PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

UMA IA PARA O JOGO COLONIZADORES DE CATAN

BRUNO PAZ E GABRIEL RUBIN

Proposta de Trabalho de Conclusão apresentada como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Felipe Meneguzzi

Porto Alegre
2016

LISTA DE SIGLAS

IA – Inteligência Artificial

IBM – *International Business Machines*

MDP – *Markov Decision Process*

MCTS – *Monte-Carlo Tree Search*

POMDP – *Partially Observable Markov Decision Processes*

SUMÁRIO

AN AI FOR THE GAME SETTLERS OF CATAN

ABSTRACT

Colonizadores de Catan é um dos principais representantes dos jogos estratégicos modernos e tem muitas características que o tornam difícil de ser jogado por uma IA. Este jogo possui poucas implementações de IA disponíveis, as quais não representam desafio para um jogador. Neste trabalho, propomos pesquisar algoritmos de jogo para projetar uma nova IA capaz de jogar Catan com uma boa performance contra outros agentes e humanos.

Keywords: .

1. INTRODUÇÃO

Jogos de tabuleiro representam um desafio para a comunidade de Inteligência Artificial. O estudo de jogos clássicos de dois jogadores e informação perfeita, como Xadrez, Damas e Go, foram importantes para o desenvolvimento da área [SCS10]. Muitas técnicas de IA foram desenvolvidas para melhorar a performance de uma IA nestes jogos clássicos, como Minimax e *Alpha-beta Pruning* [Mar87]. Apesar de lidar bem com jogos tradicionais, estas técnicas muitas vezes não são satisfatórias para jogos estratégicos modernos, comumente chamados de "eurogames", devido à maior complexidade de grande parte destes jogos quando comparada a jogos de tabuleiro tradicionais [SCS10]. Técnicas recentemente desenvolvidas para jogar jogos complexos, como Monte-Carlo *Tree Search*, que melhorou expressivamente a performance de uma IA no clássico jogo Chinês Go [GKS+12], trazem novas possibilidades de progresso no desenvolvimento de técnicas voltadas para jogos estratégicos modernos [CBSS08].

A categoria de jogos estratégicos modernos é de grande interesse para a comunidade de IA devido as características que estes jogos dividem com jogos de tabuleiro clássicos e jogos de videogame [CBSS08]. Colonizadores de Catan representa bem o arquétipo de um "eurogame", e apesar de várias implementações de IAs para este jogo existirem, nenhuma delas representa um desafio para um jogador experiente [CBSS08]. Neste trabalho, pretendemos estudar como melhorar a performance de uma inteligência artificial neste jogo com o estudo das técnicas de IA para jogos de tabuleiro clássicas, como Minimax, e modernas, como Monte-Carlo *Tree Search*, e então desenvolver uma solução capaz de competir com jogadores experientes e vencer outras soluções existentes.

2. REVISÃO BIBLIOGRÁFICA

2.1 Jogos de Tabuleiro

Jogos de tabuleiro oferecem um cenário de competição abstrata, onde dois ou mais jogadores competem para atingir um certo objetivo, permitindo avaliar a aptidão dos jogadores envolvidos para resolver problemas abstratos. Esta modalidade de jogos é objeto de estudo de diversas áreas do conhecimento, e é muito importante para a IA.

2.1.1 IA e Jogos de Tabuleiro

Jogos de tabuleiro são um dos temas mais antigos a serem estudados pela comunidade de Inteligência Artificial, contribuindo muito para o desenvolvimento da área. Em 1950, logo após computadores serem programáveis, um programa de Xadrez foi programado por Claude Shannon [Sha50]. A razão da popularidade desse tema, é que ele pode ser representado mais facilmente do que cenários não abstratos da vida real e permitem que agentes programados testem as suas habilidades contra seres humanos, avaliando assim o progresso científico atingido pela área da Inteligência Artificial.

Outra qualidade deste tema para IA, é o seu apelo ao público geral. Avanços atingidos em jogos populares, como o Xadrez, tiveram grande atenção na mídia, gerando entusiasmo em relação ao futuro da Inteligência Artificial. Episódios notórios como a vitória do *DeepBlue* da IBM contra o então campeão de Xadrez, Garry Kasparov, em 1996 [CHH02], e mais recentemente, a vitória do *DeepMind* do Google contra o campeão mundial de Go [SHM+16], são tidos como conquistas da Inteligência Artificial que marcaram a história. Acreditamos que por conta disso, este é um objeto de estudo expressivo e relevante para a comunidade.

2.1.2 Teoria de Jogos

Teoria de Jogos [FT91] é um ramo da matemática aplicada que estuda formalmente o cenário de conflito e cooperação entre agentes racionais, e também formaliza diversos aspectos de jogos como: Estratégias, tomadas de decisão e a representação formal de um jogo, servindo como base para a pesquisa deste tema no ramo da Inteligência Artificial. Iremos abordar aqui algumas das teorias básicas que serão importantes em nosso trabalho.

A Teoria de Jogos define que jogos podem ser divididos em diversas categorias de acordo com suas características [MF09], o que ajuda a classificá-los e determinar a complexidade no desenvolvimento de um agente capaz de jogá-los, seguem algumas das classificações que serão importantes para o nosso trabalho.

Jogos são determinísticos quando não possuem nenhum elemento de chance dentro de suas regras, como o rolar de dados para a definição de alguma ação. Jogos que possuem tais elementos são estocásticos. Jogos estocásticos são mais complexos pela dificuldade de se considerar diversos futuros possíveis para o jogo.

Jogos de informação perfeita são aqueles onde todas as informações do estado de jogo estão disponíveis para todos os jogadores. Jogos que possuem informações obscuras, como cartas viradas para apenas um dos jogadores, apresentam maior desafio, já que eles não podem definir com precisão o estado atual do jogo, apenas estimá-lo.

Jogos de soma-zero são aqueles onde a soma final da pontuação de todos os jogadores envolvidos é sempre zero e sempre existe um vencedor. O ganho obtido por um jogador se traduz em perda para os outros jogadores e sempre é possível medir com precisão qual dos jogadores está ganhando.

O número de jogadores também influência na complexidade de um jogo. Quanto maior o número de jogadores, maior será a árvore de jogo, já que ela deve representar estados e ações para cada jogador, o que aumenta a complexidade do problema.

Estas categorias nos ajudam a avaliar os jogos de tabuleiro e considerar quais são as melhores técnicas e algoritmos que podem ser utilizados para criar um agente capaz de jogá-los. Espera-se que um agente tenha um comportamento ótimo no jogo, ou seja, que faça sempre a melhor jogada possível em qualquer situação a fim de sair vitorioso.

2.2 Colonizadores de Catan

Colonizadores de Catan é um jogo de tabuleiro moderno criado pelo alemão Klaus Teuber e publicado em 1995. O jogo se popularizou rapidamente e ganhou notoriedade fora da Alemanha, fazendo com que o gênero conhecido como "eurogame"fosse reconhecido em todo o mundo [SCS10]. Uma das principais razões do sucesso de Catan é a sua jogabilidade: as regras e funcionamento do jogo foram desenvolvidas buscando o equilíbrio entre estratégia, política entre jogadores e sorte. Este equilíbrio de jogo permite que um iniciante tenha chances de competir contra outros jogadores mais experientes e se divertir com a partida. Além disso, o jogo tem profundidade estratégica suficiente para fomentar uma forte comunidade competitiva, que cresce desde o seu lançamento.

Existem diversas implementações de agentes que jogam Colonizadores de Catan disponíveis hoje. Podemos destacar 2 implementações como sendo as melhores [SCS10].

A primeira é proprietária da Castle Hill Studios, parte da MSN Games da Microsoft. A outra é um programa *open-source* desenvolvido por Robert S. Thomas em Java chamado JSettlers. [TH02]

2.2.1 Regras e descrição de jogo

Colonizadores de Catan possui diversas expansões, iremos descrever aqui a sua versão básica que pode ser jogada por 3 ou 4 jogadores competitivamente. O jogo possui um tabuleiro formado por hexágonos, onde cada hexágono representa um terreno que produz um tipo de recurso: a floresta produz madeira, a plantação produz trigo, a montanha produz pedra, a mina produz tijolo, o pasto produz ovelhas e o deserto não produz recurso. Existe apenas um hexágono de deserto no jogo. Estes hexágonos podem ser configurados de diversas maneiras antes do começo da partida para permitir diferentes cenários de jogo. Um número é atribuído para cada hexágono, entre 2 e 12, exceto o deserto que não produz recurso. Cada jogador possui peças de uma cor específica para diferenciá-los dentro do tabuleiro. Estas peças representam as edificações do jogador, que podem ser: Estradas, aldeias e cidades. Estas edificações podem ser colocadas na adjacência de diferentes terrenos e ficam sempre presentes no jogo após serem construídas. A Figura 2.1 ilustra uma possível configuração de tabuleiro de jogo, onde a posição e número de cada terreno já estão configuradas.

Os jogadores possuem recursos que são obtidos de acordo com os terrenos do tabuleiro e podem ser trocados entre jogadores ou com o banco de jogo. Cercando os terrenos do jogo, existe o oceano, que possibilita aos jogadores com edificações próximas usufruir de diferentes regras para troca. O jogo ainda possui 2 dados, diversas cartas que representam os recursos de um jogador e um baralho de cartas de evento que podem ser compradas e utilizadas pelos jogadores.

Colonizadores de Catan é uma corrida por pontos entre os jogadores. Aquele que obtiver 10 pontos de vitória, vence o jogo. Para obter pontos, os jogadores devem construir edificações no tabuleiro, comprar cartas de desenvolvimento ou atingir 2 objetivos de jogo. Aldeias valem 1 ponto de vitória, cidades valem 2 e existem cartas de desenvolvimento que valem 1 ponto, já os objetivos valem 3 pontos cada. Um objetivo é ser o primeiro jogador a obter 3 cartas de cavalaria e o outro é ter a maior estrada construída durante a partida.

Figura 2.1 – Tabuleiro do jogo

Uma partida começa com uma configuração inicial para cada jogador dentro do tabuleiro. Na configuração para iniciantes, cada um começa com 2 aldeias e 2 estradas construídas, ou seja, 2 pontos de vitória, e algumas cartas de recurso. Os jogadores então jogam em turnos até que um deles atinja 10 pontos de vitória. No começo de seu turno, o jogador deve rolar 2 dados de 6 lados para definir quais terrenos do tabuleiro irão produzir recursos. Se o número obtido nos dados for 7, o jogador pode mover o ladrão, uma peça especial de jogo.

O ladrão inicia a partida acima do deserto e está sempre presente no jogo. Ele pode ser visto na Figura 2.1 em sua posição inicial. Esta peça pode ser movida com o uso da carta de cavalaria por algum dos jogadores ou com os dados, como descrito anteriormente. O terreno em que o ladrão estiver não produz recursos. Isso pode ser utilizado para "negar"a produção de recursos de um jogador adversário. O jogador que mover o ladrão para um terreno que "nega"a produção de recurso de algum outro jogador, pode roubar uma carta de recurso deste jogador escolhida aleatoriamente.

Após rolar os dados, os jogadores podem então trocar recursos entre si ou com o banco de jogo. Trocas entre jogadores podem ser feitas com taxas de conversão definidas por eles mesmos, já trocas com o banco só podem ser feitas com uma taxa de 3:1, exceto se o jogador tiver uma edificação próxima ao oceano do jogo, então taxas especiais para troca com o banco se aplicam. Depois de realizar as trocas, o jogador pode então investir seus recursos para construir edificações e comprar cartas de desenvolvimento. Existem mais algumas regras específicas para a construção de edificações no tabuleiro, além de compra e uso de cartas de desenvolvimento que cada jogador deve respeitar em seu turno, mas elas não alteram este fluxo básico de jogo.

2.2.2 Características e desafios

Colonizadores de Catan tem três características que o tornam complexo quando comparado aos jogos de tabuleiro clássicos [SCS10]. Esta modalidade de jogos geralmente é determinística, observável e de 2 jogadores [CBSS08]. Catan, por outro lado, é estocástico, por necessitar da rolagem de dados durante os turnos, parcialmente observável, por conter cartas de desenvolvimento que ficam abertas apenas para o jogador que as possui, e tem mais de 2 jogadores [SCS10].

Outro aspecto do jogo, que é bastante desafiador para agentes, é o fato de jogadores poderem trocar recursos livremente entre si, com o banco de jogo ou com um porto que estejam adjacentes. Levando em conta esse sistema de troca e a quantidade de jogadores da partida, o agente deve estar preparado para tomar decisões políticas e decidir entre fazer ou não troca de recursos com adversários ou com o banco de jogo.

2.3 Técnicas de Jogo para IA

Esta seção apresenta algumas das técnicas de IA que são comumente aplicadas em agentes que jogam jogos de tabuleiro clássicos e servem de modelo para técnicas de IA modernas. Estudaremos estes algoritmos para modelar nosso agente que jogará Catan.

2.3.1 Minimax

O Minimax é um algoritmo que serve de base para grande parte dos algoritmos de busca em árvore de jogo [vdW05]. Composto por um conjunto de estados, jogadas e uma função de utilidade, para calcular quanto vale cada estado final do jogo, sua função é

encontrar a estratégia ótima, a fim de vencer a partida, construindo uma árvore de jogo que modela a competição entre dois jogadores em um jogo de soma-zero da seguinte forma:

- De um certo estado do jogo, o algoritmo cria toda a árvore de jogo, e calcula a utilidade dos estados finais da árvore utilizando a função de utilidade.
- Partindo do estado final, a utilidade de cada estado intermediário do jogo é calculado subindo pela árvore, sempre trocando entre uma etapa de maximização de utilidade e outra de minimização, para simular a jogada de ambos os jogadores, partindo do pressuposto de que ambos vão sempre buscar as jogadas ótimas, que possuem maior utilidade.
- Finalmente, quando a raiz é atingida, a maior utilidade é escolhida dentre os estados imediatamente abaixo do estado raiz e uma jogada é feita pela máquina.

Uma possível aplicação deste algoritmo é no jogo Tic-Tac-Toe(Jogo da Velha). Apesar de ser um jogo muito simples e com poucos recursos, a árvore gerada pelo algoritmo é de certa forma bastante profunda, visto que esta árvore representa todas a jogadas possíveis do jogador (MAX) com base nas jogadas do o oponente (MIN). Ao final da árvore, as folhas são os possíveis resultados de jogo, vitória, empate ou derrota são avaliados com um valor de utilidade, sendo a vitória de MAX o mais útil, conforme a Figura 2.2.

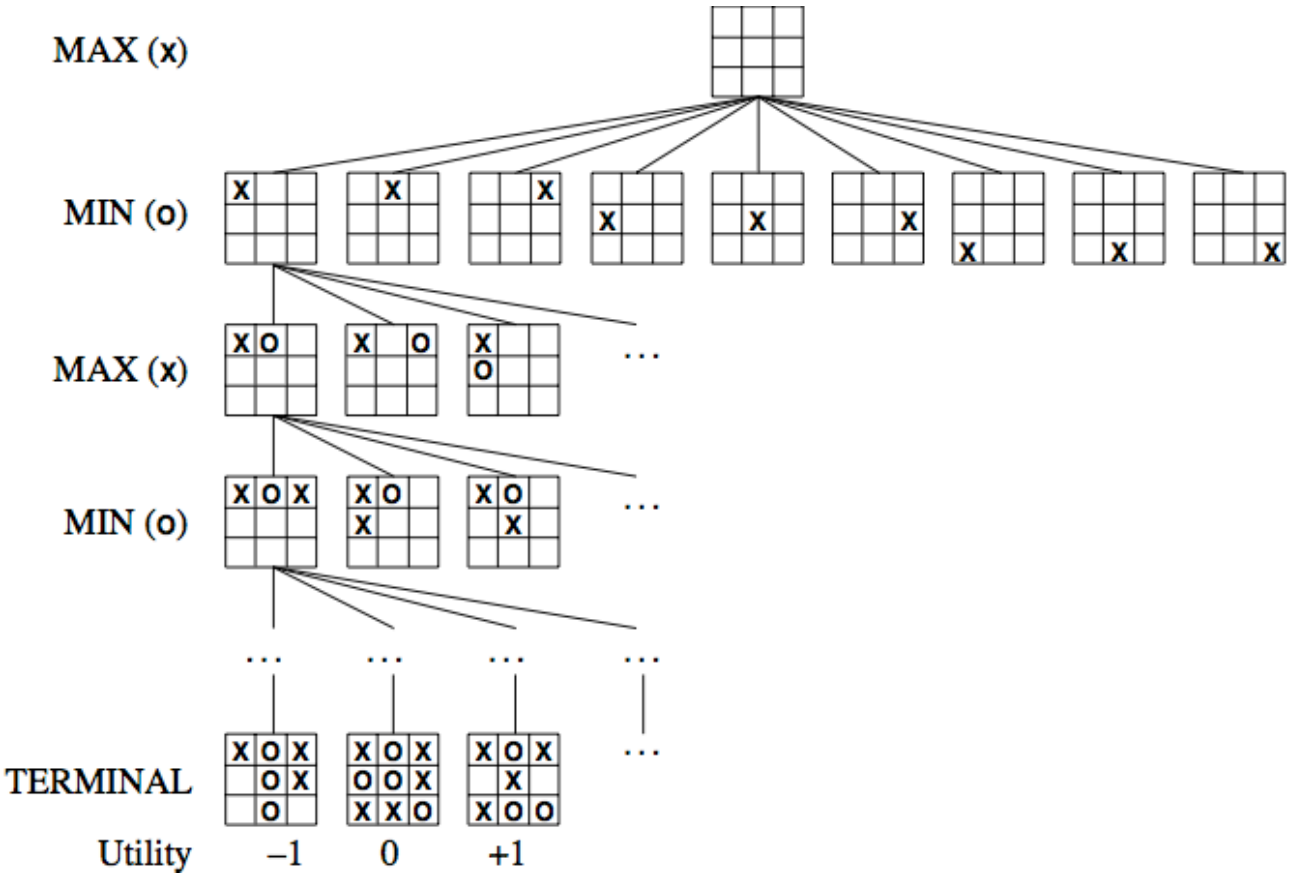

Figura 2.2 – Apenas uma parte da árvore gerada pelo algoritmo Minimax. Para cada ação de MIN uma nova ramificação é gerada para MAX. Ao final da árvore as folhas são o resultado do jogo, e um valor de utilidade é atribuído para cada resultado. [RN10, Cap 5, pp163-166]

Este algoritmo serve de modelo ideal para resolver diversos jogos, porém é bastante custoso e complexo, devido ao seu fator de ramificação alto, ou seja, a árvore de possibilidades de jogadas cresce rapidamente e torna vários jogos intratáveis para um computador. Afim de evitar o crescimento da árvore gerada, algumas técnicas de corte foram elaboradas. A ideia é descartar nodos, eliminando ramificações desnecessárias, reduzindo o problema e aumentando a performance do algoritmo.

2.3.2 Técnicas de corte para Minimax

Alpha-Beta Pruning é uma técnica que pode ser utilizada para desconsiderar alguns nodos da árvore que não irão influenciar na escolha da jogada. Devido à natureza do Minimax de maximizar uma etapa e minimizar a seguinte, alguns nodos da árvore não são relevantes durante o processo de seleção e, utilizando esta técnica, podemos descartar eles mesmo antes de considerar-los como é mostrado na Figura 2.3. Conforme o caminhamento na árvore, são definidos dois valores, α e β:

- α = é o valor do melhor caminho já explorado até a raiz para o jogador (MAX).

- β = é o valor do melhor caminho já explorado até a raiz para o oponente (MIN).

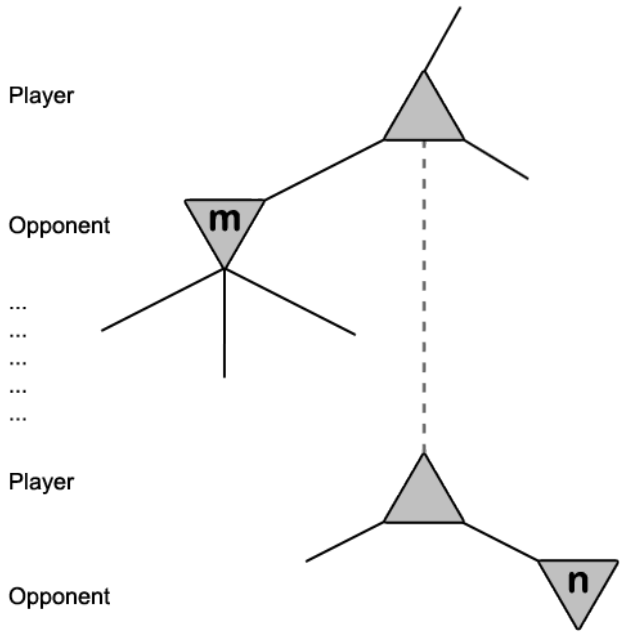

Figura 2.3 – A figura acima mostra um exemplo do comportamento do *pruning*. Se o valor atribuído da jogada de MIN *m* for melhor que o valor de um possível resultado de partida *n*, então a partida nunca irá resultar em *n*, por ser um resultado ruim para MAX. [RN10, Cap 5, pp167-171]

Mesmo com a redução de possibilidades de jogadas que seriam ruins, a técnica *Alpha-Beta Pruning* ainda precisa percorrer grande parte da árvore de jogo, o tornando não muito eficiente para jogos com uma certa complexidade. Uma solução para este problema é substituir a função de utilidade por uma Função de Estimação heurística. Uma Função de Estimação ou *Evaluation Function* tem como objetivo estimar um valor para determinado ponto do jogo, ou seja, ao invés do algoritmo percorrer todo caminho até um nodo final, ele simplesmente pode parar no meio do caminho e estimar um valor com base no que já foi percorrido. Este valor é estimado melhor quando há uma possível chance de vitória. [RN10, Cap 5, pp171-175]

2.3.3 Expectimax

A técnica *Alpha-Beta Pruning* reduz visitas em nodos, mas ainda assim percorre grande parte da árvore de jogo. Uma forma de otimizar esta técnica é usar uma *Evaluation Function*, assim o caminho percorrido será muito menor, visto que ao chegar em um certo nodo, não terminal, será estimado um valor para o mesmo, e com base nos nodos percorridos anteriormente, assumir que este nodo é terminal.

Jogos como o Catan apresentam elementos de chance, ainda não considerados nas técnicas vistas acima. O algoritmo Expectimax lida com este problema, introduzindo uma nova etapa, de nodos de chance na árvore de jogo, como mostra a figura 2.4, além dos nodos de Mínimo e Máximo.

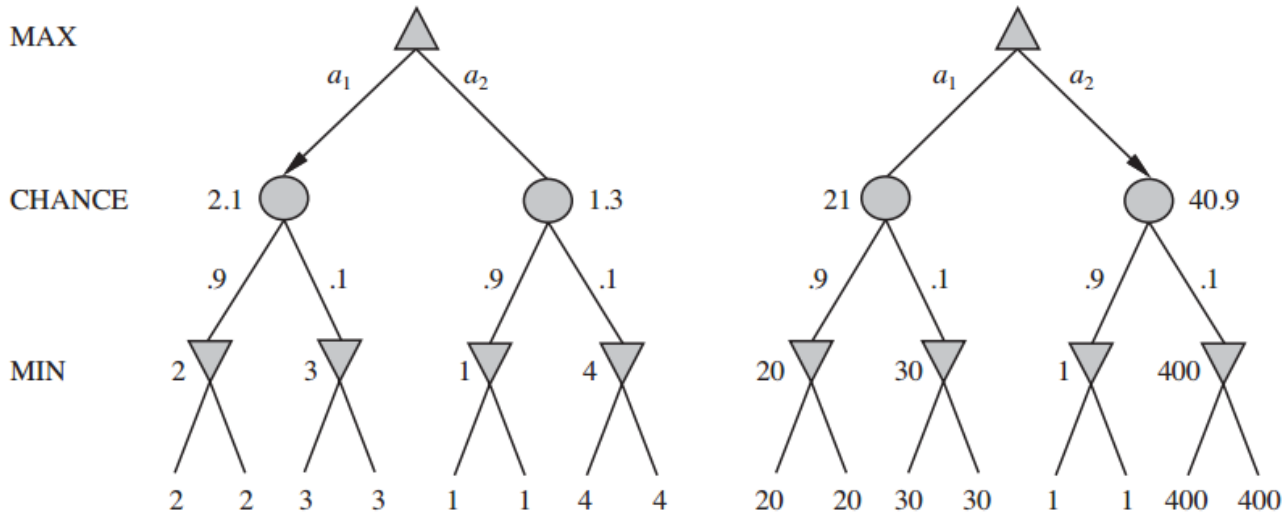

Figura 2.4 – Os nodos de chance no jogo Catan, por exemplo, seriam as possibilidades ao jogar os dados. [RN10, Cap 5, pp177-180]

2.4 Tomada de Decisão em Ambientes Estocásticos

2.4.1 *Markov Decision Process* (MDP)

Um MDP busca encontrar a política que garanta a maior recompensa em um sistema de decisões sequenciais, observável e estocástico, definindo uma coleção de estados, ações, um modelo de transição e uma função de recompensa. Os estados representam diferentes situações do problema e as ações definem transições entre estados. O modelo de transição determina a probabilidade de se atingir um certo estado x' se uma ação a é aplicada à partir de um estado x. A função de recompensa define a utilidade de cada estado para o agente. A política é um mapeamento de estados para ações que define qual ação deve ser escolhida partindo de um determinado estado.

Se o sistema de decisões for parcialmente observável, o problema se torna mais complexo. *Partially Observable Markov Decision Processes* (POMDPs) é o nome da classe de problemas que possui essa característica. Como Catan é parcialmente observável devido as cartas de desenvolvimento e as trocas aleatórias de recursos entre jogadores, o agente não tem o modelo de transição e se encaixa nesta categoria. Uma abordagem para

resolver POMDPs é calcular a distribuição de probabilidade de todos os estados do problema com base nas decisões anteriores, e a partir deste cálculo, reduzir o problema em um MDP para esta distribuição. [RN10, Cap 17, pp645-647]

2.4.2 Aprendizado por Reforço

Aprendizado por Reforço é uma técnica de aprendizado onde um agente aprende o que fazer diante de um problema a partir de recompensas obtidas ao tentar resolvê-lo repetidas vezes, durante um treinamento [RN10, Cap 21, pp830-859]. Recompensas, neste caso, são as mesmas vistas em MDPs, e o objetivo do agente ainda é encontrar uma política ótima para lidar com um problema. Em um jogo como Catan, as recompensas podem ser vistas como ganhar um ponto de vitória, vencer a partida ou perder a partida, o último seria um estimulo negativo para o agente.

Diferente de outros métodos de aprendizado, este não necessita de supervisão humana. Em aprendizado supervisionado, por exemplo, um humano informa ao agente qual é a ação correta a ser tomada a partir de um estado e o agente aprende com estas orientações [RN10, Cap 18, pp695-697]. Por conta disso, aprendizado por reforço é mais adequado para jogos de tabuleiro, onde informar o agente sobre o que fazer para todos os estados de jogo pode ser impraticável. Sem informações prévias sobre todos os estados e resultados de ações, ou seja, o modelo de transição da MDP, o agente deve aprender a lidar com uma MDP desconhecida.

2.4.3 Exploração e Lucro

Exploração e lucro são conceitos chave no aprendizado por reforço [RN10, Cap 21, pp830-859]. Visto que o agente deve aprender o modelo de transição sem ter conhecimento do modelo real do ambiente, se ele priorizar o lucro imediato, ele poderá estar aprendendo um modelo equivocado, que não representa o problema. Para que o agente aprenda o modelo de transição ótimo, ele deve também considerar ações pouco lucrativas a fim de conhecer mais sobre o modelo de transição real do ambiente.

O agente deve buscar então uma estratégia de exploração ótima. Essa estratégia deve explorar o problema para obter o máximo de aprendizado em um primeiro momento e depois passar para uma fase de lucro, explorando mais profundamente os nodos que geraram mais lucro, de modo que o modelo aprendido pelo agente seja ótimo com o menor esforço de exploração possível.

2.4.4 Problema das N-Máquinas Caça-Níqueis

O problema das N-máquinas caça-niqueis é estudado na área de decisões estatísticas e busca por uma política de exploração ótima [BF79]. Se encararmos problemas de decisão como a escolha de qual máquina caça-níquel paga mais dentre diversas máquinas independentes em um exato momento, podemos descobrir a melhor política de exploração dentre as máquinas de acordo com a a distribuição de lucros observada depois de investir em todas as máquinas repetidas vezes. Pode não ser possível investir em todas as máquinas o número de vezes necessário para se descobrir a melhor política, mas é possível garantir uma política que converge para a política ótima conforme o número de investimentos sobe, e pode-se chegar em uma aproximação satisfatória dentro do limite de simulação disponível.

2.4.5 *Q-Learning*

Dentre as técnicas de aprendizado por reforço, *Q-Learning* se diferencia das outras por ser uma técnica livre de modelo de transição [WD]. Em *Q-Learning* o agente aprende um valor de recompensa associado à uma tupla ação-estado que converge para resultados similares ao de se aprender o modelo de transição, o que significa que o agente necessita de menos conhecimento do domínio, simplificando o processo de aprendizado, porém restringindo o seu potencial de aprender em ambientes complexos, já que ele não poderá simular os resultados das ações possíveis. Enquanto outras técnicas visam aprender o modelo de transição da MDP desconhecida, esta técnica não necessita do modelo para aprender a resolver o problema.

2.5 *Monte-Carlo Tree Search* (MCTS)

Monte-Carlo Tree Search é uma família algoritmos modernos para resolver problemas, principalmente jogos, que não exige conhecimento do domínio do problema para obter resultados satisfatórios [BPW+12]. Uma vantagem desse algoritmo em relação ao Minimax é que ele pode ser usado em jogos estocásticos e parcialmente observáveis, enquanto o Minimax necessita de adaptações para cumprir tais requisitos.

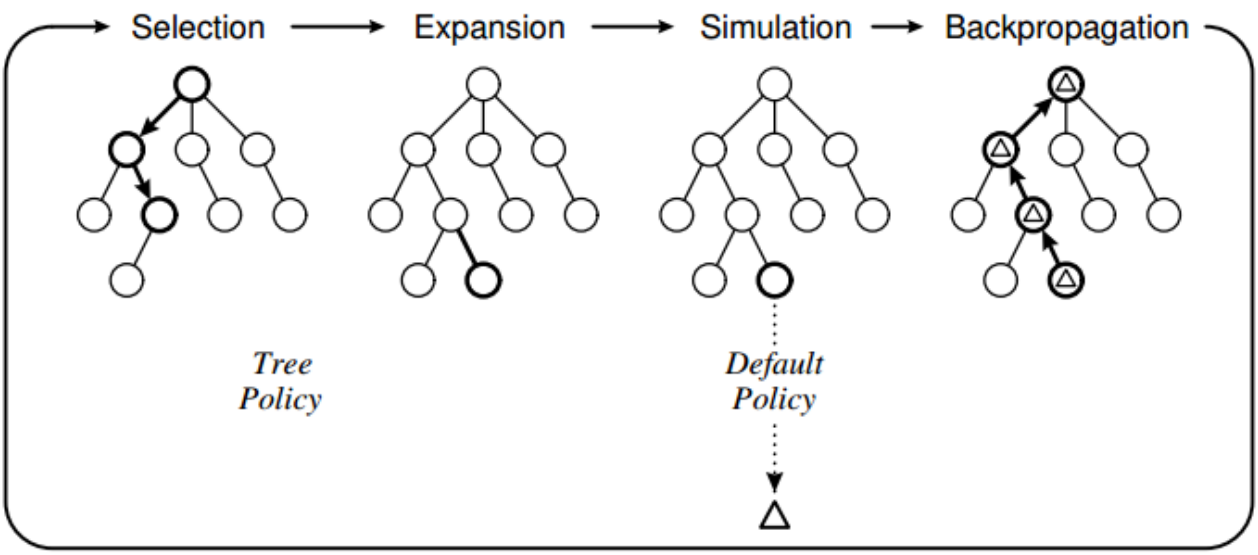

Figura 2.5 – Etapas da Àrvore de Pesquisa de Monte-Carlo

O algoritmo funciona em 4 estágios, de acordo com a Figura 2.5. A partir de um nodo raiz, o algoritmo expande os próximos nodos possíveis. Cada nodo possui uma informação estatística que ajuda o algoritmo a decidir qual será o próximo nodo a ser explorado. Seguem as descrições de cada estágio do algoritmo:

- Seleção: Do nodo raiz, é selecionado um nodo inexplorado da árvore de jogo a partir de uma politica de seleção, que escolhe a jogada mais promissora buscando lucrar ou explorar.
- Expansão: Um ou mais nodos são adicionados como filhos do nodo selecionado a partir das jogadas válidas possíveis.
- Simulação: O jogo é simulado a partir de um ou mais nodos gerados na fase de expansão seguindo uma política de jogo padrão, que pode ser movimentos aleatórios, semi-aleatórios ou algo mais sofisticado. Quanto mais sofisticada a política padrão, mais custosa e lenta será a simulação.
- Propagação Inversa: Com a recompensa obtida na fase de Simulação, a informação estatística de cada nodo é atualizada, partindo do nodo adicionado até a raiz.

Com uma quantidade suficiente de simulações, a melhor jogada pode ser inferida a partir de uma política de jogo padrão simples, ou até mesmo aleatória, devido a experiência obtida pela propagação dos resultados da simulação na árvore [BPW+12]. Uma função de exploração é essencial para os algoritmos de Monte-Carlo e dentre os mais populares e efetivos, estão os métodos que se baseiam no problema das N-máquinas caça-niquel.

Uma MCTS pode ser vista como um tipo de aprendizado por reforço [BPW+12]. Ambas as técnicas utilizam uma amostragem de simulações para construir conhecimento

acerca do problema. Sob certas circunstâncias, ambos os algoritmos podem até ser equivalentes [Sil09], mas algoritmos de aprendizado por reforço, como *Q-Learning*, geralmente não constroem árvores. Outra diferença é que para decidir qual será a próxima ação, MCTS estima valores temporários para cada estado, enquanto aprendizado por reforço, como *Q-Learning*, estima valores a longo prazo, que então ajudarão o agente a tomar decisões futuras [BPW+12].

3. OBJETIVOS

Este trabalho tem como objetivo desenvolver uma IA capaz de jogar o jogo Colonizadores de Catan e medir a sua capacidade de jogo.

3.1 Objetivo Geral

Projetar uma IA capaz de jogar Colonizadores de Catan, utilizando o conhecimento adquirido no estudo do cliente do jogo e das técnicas de IA descritas nesta proposta.

3.2 Objetivos Específicos

- Compreender e modelar formalmente o jogo Colonizadores de Catan, com o estudo do cliente de jogo *JSettlers*.
- Entender algoritmos clássicos baseados no Minimax.
- Compreender como os algoritmos de Aprendizado por Reforço podem ajudar o agente a aprender as regras de jogo e sua relação com MCTS.
- Entender o funcionamento de técnicas de MCTS e escolher aquelas que serão importantes no Colonizadores de Catan.
- Descobrir outras técnicas de IA capazes de melhorar a performance de um agente no jogo.
- Implementar um pequeno protótipo para um ou mais algoritmos de IA descritos nesta proposta.
- Projetar um agente capaz de jogar Colonizadores de Catan utilizando um ou mais dos algoritmos estudados.

4. ANÁLISE E PROJETO

Afim de atingir os objetivos, será necessário a realização de algumas atividades como, entender as estratégias do jogo Colonizadores de Catan e técnicas de IA que podem ser aplicadas para resolve-lo. O detalhamento destas atividades será visto a seguir.

4.1 Atividades

- Estudar o cliente *JSettlers*: O cliente *JSettlers* é uma implementação em Java do jogo Colonizadores de Catan. O estudo irá contribuir para a implementação da IA que irá jogar neste cliente.
- Estudar o algoritmo Minimax: O Minimax é um algoritmo voltado para IA em jogos. Seu funcionamento, aplicando algumas técnicas, pode ser aproveitado na realização da IA que joga Catan.
- Estudar tomada de decisão: Há diversos algoritmos que buscam encontrar a política ótima em um sistema de decisões sequenciais, fazendo assim o agente realizar uma ação apropriada para o momento. Fundamental para o entendimento dos algoritmos estudados.
- Estudar aprendizado por reforço: O aprendizado por reforço utiliza a noção de recompensas para ensinar agentes como se comportar diante de um problema. Esta abordagem pode ser aproveitada como uma alternativa para o agente aprender sobre o jogo proposto.
- Estudar *Monte-Carlo Tree Search*: Este é um algoritmo moderno que teve muito sucesso quando aplicado a diversos jogos. Ele utiliza uma amostragem de simulações para construir conhecimento acerca de um problema. O estudo de técnicas baseadas neste algoritmo podem contribuir na criação da IA que jogará Catan.
- Pesquisar outras técnicas: Estudaremos também outras técnicas que podem ser utilizadas para melhorar a performance da IA.
- Modelagem do agente: Será realizada a modelagem de um agente capaz de jogar o jogo no cliente *JSettlers* que possa utilizar mais de uma das técnicas estudadas.

4.2 Cronograma

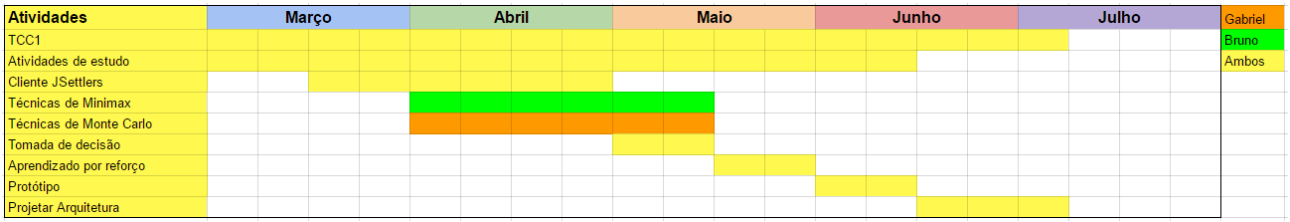

Figura 4.1 – Cronograma das atividades planejadas.

REFERÊNCIAS BIBLIOGRÁFICAS

[BF79] Berry, D. A.; Fristedt, B. “Bernoulli one-armed bandits–arbitrary discount sequences”, *Ann. Statist.*, vol. 7–5, 09 1979, pp. 1086–1105.

[BPW+12] Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. “A survey of monte carlo tree search methods”, *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4–1, March 2012, pp. 1–43.

[CBSS08] Chaslot, G.; Bakkes, E.; Szita, I.; Spronck, P. “Monte-carlo tree search: A new framework for game ai”. In: In Proceedings of AIIDEC-08, 2008, pp. 216–217.

[CHH02] Campbell, M.; Hoane, Jr., A. J.; Hsu, F.-h. “Deep blue”, *Artif. Intell.*, vol. 134–1-2, Jan 2002, pp. 57–83.

[FT91] Fudenberg, D.; Tirole, J. “Game Theory”. Cambridge, MA: MIT Press, 1991, translated into Chinesse by Renin University Press, Bejing: China.

[GKS+12] Gelly, S.; Kocsis, L.; Schoenauer, M.; Sebag, M.; Silver, D.; Szepesvári, C.; Teytaud, O. “The grand challenge of computer go: Monte carlo tree search and extensions”, *Commun. ACM*, vol. 55–3, Mar 2012, pp. 106–113.

[Mar87] Marsland, T. A. “Computer chess methods”. In: Shapiro, S., ed.: Encycploedia of Artificial Intelligence, 1987, pp. 157–171.

[MF09] Millington, I.; Funge, J. “Artificial Intelligence for Games, Second Edition”. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, 2nd ed..

[RN10] Russell, S. J.; Norvig, P. “Artificial Intelligence: A Modern Approach (Third Edition)”. Upper Saddle River, New Jersey, USA: Prentice Hall, 2010.

[SCS10] Szita, I.; Chaslot, G.; Spronck, P. “Monte-carlo tree search in settlers of catan”. In: Proceedings of the 12th International Conference on Advances in Computer Games, 2010, pp. 21–32.

[Sha50] Shannon, C. E. “Xxii. programming a computer for playing chess”, *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 41–314, 1950, pp. 256–275, http://dx.doi.org/10.1080/14786445008521796.

[SHM+16] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; Hassabis, D. “Mastering the game of go with deep neural networks and tree search”, *Nature*, vol. 529, 2016, pp. 484–503.

[Sil09] Silver, D. “Reinforcement learning and simulation-based search in computer go”, Tese de Doutorado, Edmonton, Alta., Canada, 2009, aAINR54083.

[TH02] Thomas, R.; Hammond, K. “Java settlers: A research environment for studying multi-agent negotiation”. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, 2002, pp. 240–240.

[vdW05] van der Werf, E. “AI Techniques for the Game of Go”. UPM, Universitaire Pers Maastricht, 2005.

[WD] Watkins, C. J. C. H.; Dayan, P. “Q-learning”, *Machine Learning*, vol. 8–3, pp. 279–292.