UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ROGIEL JOSIAS SULZBACH

Método de classificação e comparação de build orders em StarCraft II

Porto Alegre

2016

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ROGIEL JOSIAS SULZBACH

Método de classificação e comparação de build orders em StarCraft II

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Altamiro Amadeu Susin

Porto Alegre
2016

Lista de Figuras

Figura 1 – Exemplo de uma distribuição de probabilidade de uma ação com 2 repetições 9
Figura 2 – Exemplo da distribuição de probabilidade da ação A_1. A curva em azul representa a primeira repetição da ação (nos tempos t_1 e t_2), a curva em laranja representa a segunda repetição (nos tempos t_3 e t_4) e a curva em verde representa a terceira repetição da ação (tempos t_6 e t_7). . . . 16
Figura 3 – Exemplo da distribuição de probabilidade da ação A_1. Como a ação A_2 somente é executada uma única vez no exemplo, apenas uma repetição é mostrada no gráfico. A ação corresponde aos instantes de tempo t_5 e t_8. 17

Lista de Tabelas

Tabela 1 – Um exemplo de uma *build order* com as ações A_1 e A_2 ocorrendo nos tempos t_n. 14

Lista de Abreviaturas e Siglas

BO	*Build Order*
SC2	*StarCraft II*
PDF	*Probability Distribution Function*
CDF	*Cumulative Distribution Function*
RTS	*Real Time Strategy*

Sumário

1 INTRODUÇÃO 6
1.1 Sobre o StarCraft II 6
1.2 O uso de aprendizado de máquina em jogos de estratégia em tempo real 7

2 REVISÃO BIBLIOGRÁFICA 8
2.1 Trabalhos anteriores 8
2.2 A estrutura estatística de uma *build order* 9
2.3 Fator de *branching* de um jogo de StarCraft: BroodWar 10
2.4 O Teorema de Bayes 10
2.5 Redes Bayesianas 11

3 MÉTODO 12
3.1 Extração dos dados 12
3.2 *Dataset* de referência 13
3.3 Treinamento 14
3.4 Classificação 16

4 RESULTADOS E DISCUSSÕES 18

5 CONCLUSÕES 19

6 PROPOSTAS DE TRABALHOS FUTUROS 20

REFERÊNCIAS BIBLIOGRÁFICAS 21

1 Introdução

1.1 Sobre o StarCraft II

StarCraft II é um jogo de estratégia militar em tempo-real desenvolvido pela *Blizzard Entertainment* onde três facções disputam partidas multiplayer em modalidade 1 contra 1. O objetivo do jogo consiste em conseguir destruir todas as estruturas do adversário que provém infraestrutura para a construção e manutenção do exército.

O jogo possui um sistema de economia, onde para que um jogador possa treinar ou desenvolver uma tecnologia, é necessário que primeiro sejam extraído recursos do ambiente virtual de jogo. O jogo também apresenta um sistema de "árvore tecnológica"onde há um encadeamento de pré-requisitos para o treinamento e construção de unidades de exército mais avançadas. É a existência desta árvore tecnológica que possibilita que seja possível inferir e prever uma determinada estratégia do jogador.

O modo mais comum de execução de uma estratégia é utilização de *build orders*. Uma *build order* é uma sequência de ações tomadas por um jogador no decorrer do jogo, que, quando executadas de forma correta, oferecem alguma vantagem estratégica para o jogador. Muitas build orders são padronizadas e otimizadas por jogadores profissionais durante o treino e são popularizadas em campeonatos mundiais.

Neste trabalho será desenvolvido e implementado um método de classificação para *build orders* de partidas de jogadores profissionais de StarCraft II. Foi feita a escolha de utilizar partidas profissionais pois são jogadores com elevado conhecimento do jogo e reagem de forma ótima para várias situações inusitadas ou inesperadas, o que reduz a variabilidade das medidas extraídas dos arquivos de *replay* do jogo. Para a extração de dados foi utilizado um pacote de *replays* (arquivo binário que codifica todas as ações tomadas em um jogo) de competições profissionais de StarCraft II durante o ano de 2016.

Uma aplicação direta para o método proposto é o desenvolvimento de uma inteligência artificial que seja capaz de tomar decisões ao longo de um jogo de forma verossímil à um jogador humano, conforme proposto em (SYNNAEVE; BESSIERE, 2011a).

1.2 O uso de aprendizado de máquina em jogos de estratégia em tempo real

O uso de aprendizado de máquina em jogos de estratégia em tempo real (RTS) pode ser dividido em vários problemas: táticas, estratégias e *micro management* conforme definido por (SYNNAEVE; BESSIERE, 2011b).

Táticas se refere ao posicionamento de unidades no mapa e é diretamente depende da forma com a qual as unidades interagem entre si no jogo. Dessa forma, uma solução para este problema deve considerar o mapa e as interações entre as mecânicas de cada unidade.

Os problemas de estratégia se referem ao plano geral de jogo. A estratégia surge de uma expectativa para o desenvolver do jogo, por exemplo, supondo que um jogador deseja investir fortemente na sua economia para que consiga construir unidades tecnológicas de maior custo. A estratégia é a forma com a qual ele vai conseguir atingir este objetivo. Em geral, a estratégia é independente do mapa em que jogo está sendo jogado, mas é diretamente relacionado à raça do oponente, suas escolhas tecnológicas e sua gerência da economia (geralmente denominado de *macro management*). A estratégia de um jogador está diretamente relacionada com a sua escolha *build order*.

Por fim, os problemas de *micro management* consistem na gerência de unidades individuais, ou seja, o jogador faz o controle de cada unidade do exercito de forma individual. O *micro management* é uma especialização das táticas, mas se refere à cada unidade individual ao invés do exército inteiro.

2 Revisão Bibliográfica

2.1 Trabalhos anteriores

"A Data Mining Approach to Strategy Prediction"(WEBER; MATEAS, 2009) apresentaram uma forma de expressar uma *build order* de StarCraft Brood War em um vetor de *features* para processamento bem como a performance de quatro algoritmos de classificação. No trabalho concluíram que os diferentes algoritmos possuiam performance diferenciada nos diversos estágios de jogo. O modelo de codificação de features proposto baseava-se num critério de primeira-aparição, isto é, o vetor de *features* contém o instante em que a primeira instância de uma dada unidade ou estrutura ocorria no jogo e, caso não houvesse ocorrência, utilizava um valor padronizado de zero.

Adicionalmente, a fim de simular efeitos de *scouting* ruído foi adicionado nos vetores de *features* como forma de gerar variação nos *timings* de cada ação e então analisar a performance dos algoritmos. Foi concluído que todos os algoritmos perdiam precisão proporcional à quantidade de ruído adicionada, no entanto o algoritmo de k-NN (k-*nearest neighboors*) não degradou a performance de forma tão drástica quanto os outros algoritmos. Também foi adicionado outro teste de ruído para simular informação incompleta: lacunas foram inseridas no vetor de *features* a fim de emular a situação em que o jogador não consegue extrair uma informação do jogo do oponente. Com este tipo de ruído, a conclusão foi de que a precisão dos algoritmos de decaía de forma linear com o nível de ruído adicionado no vetor.

Em "A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft"(SYNNAEVE; BESSIERE, 2011a), baseado no trabalho anterior de (WEBER; MATEAS, 2009), realizaram o desenvolvimento de um modelo Bayesiano para classificação de build orders de StarCraft: Brood War utilizando um método de extração de *features* de ações significativas dos replays. A extração do replay considerava estruturas na arvore de tecnologia e considerava apenas a primeira construção de uma estrutura e o índide da sequencia de construção era utilizado para treinar o sistema. O método era capaz de identificar a abertura do jogador após a construção de, na média, 10 estruturas. Contudo, devido ao numero de limitado de *features*, era possível enumerar todas as possibilidades de jogos possíveis (em torno de 1000 por raça). O método era adequado para classificação em que a informação era limitada, por exemplo, era adequada para o contexto de uma inteligência artificial de toma suas decições baseada na sua capacidade de *scouting* (tentar descobrir a estratégia de um outro jogador utilizando uma unidade que descobre cada estrutura feita pelo oponente).

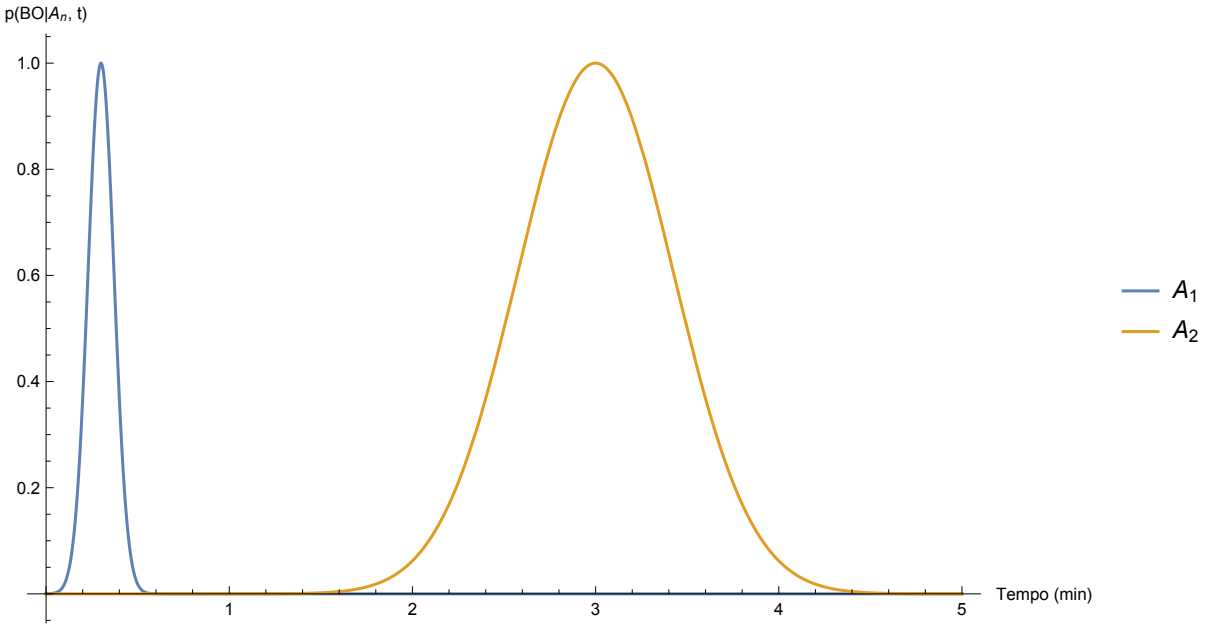
(SYNNAEVE; BESSIERE, 2011b) apresenta uma análise extensiva para técnicas e dificuldades encontradas ao aplicar algoritmos de aprendizado de máquina em diversos tipos de jogo, incluindo jogos de estratégia em tempo real (RTS).

2.2 A estrutura estatística de uma *build order*

Uma build order genérica pode ser expressa como uma sequência de ações cujo valor tempo de execução médio e desvio padrão estão relacionados com uma função distribuição de probabilidade.

Seja A uma ação arbitrária, A_n a sequência da ação durante o jogo (primeira vez que ela é executada, segunda, terceira, etc...), BO seja uma *build order* em que a sequência de ações A_n foram executadas e t o tempo de jogo em minutos. Dessa forma, podemos expressar a distribuição de probabilidade de uma sequência da forma indicada na Figura 1:

Figura 1 – Exemplo de uma distribuição de probabilidade de uma ação com 2 repetições



Na Figura 1 é possível observar que a primeira vez que a ação é executada, a tolerância para atraso é pequena, contudo, na segunda execução a tolerância de atraso é muito maior.

A tolerância de execução se faz necessária pois um jogador pode atrasar alguns segundos por múltiplos motivos durante o jogo, embora ele ainda esteja executando a mesma build-order.

Para uma *build order* real, haverá uma distribuição de probabilidade para cada uma das ações que podem oscilar em torno de 30 a 40 dependendo da facção do jogador.

2.3 Fator de *branching* de um jogo de StarCraft: BroodWar

Em teoria dos jogos , *branching factor* é o número de nós-filho em cada um dos nós referencia de ações possíveis no jogo, isto é, é o número de opções que um jogador tem disponível em cada momento.

Infelizmente não há análises cientificas feitas para a complexidade de um jogo de StarCraft II, mas é possível realizar uma comparação utilizando o jogo antecessor da série, StarCraft: Brood War lançado em 1998. De acordo com (WEBER; MATEAS, 2009), StarCraft: BroodWar possui um fator de branching estimado médio de 1×10^6 que, comparado à um jogo de xadrez onde a média é de 35, é um valor extremamente alto. O alto nível de complexidade força que seja utilizado um modelo probabilístico ao invés de uma análise determinista, uma vez que é impossível obter, ou sequer gerar, uma amostra para todas as possibilidades de jogos. Devido a novas mecânicas de jogo introduzidas em StarCraft II espera-se a complexidade seja superior à de seu antecessor.

2.4 O Teorema de Bayes

O Teorema de Bayes relaciona a probabilidade de um evento associado à uma restrição. A definição matemática do teorema é dada pela Equação **??**:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

onde A é um evento, B é a restrição do evento A. Dessa forma, é possível determinar a probabilidade do evento A ter acontecido, dado que o evento B foi observado.

Em termos da análise de *build orders*, pode-se aplicar o teorema da seguinte forma:

$$P(BO|A,t) = \frac{P(A|BO,t)P(BO)}{P(A,t)} \tag{2.2}$$

onde BO é uma *build order* qualquer que deseja-se estimar a probabilidade de estar sendo executada, dado que o jogador executou a ação A no tempo t. Para tanto, precisa-se determinar 3 fatores:

- $P(A,t)$: a probabilidade de um jogador executar a ação A no instante t, independe da *build order*;
- $P(BO)$: a probabilidade de um jogador executar a *build order* BO, independente da ação que o jogador está executando;
- $P(A|BO,t)$: a probabilidade de ter executado a ação A supondo que o jogador está executando a *build order* BO no instante t;

2.5 Redes Bayesianas

Uma rede Bayesiana (RUGGERI; KENETT; FALTIN, 2007) é um modelo probabilístico onde uma série de eventos, com suas probabilidades condicionais, representado utilizando um grafo acíclico. Um grafo acíclico é um grafo onde não há dependências cíclicas.

referencia

explicar mais

Dessa forma, uma *build order* é vista como uma rede Bayesiana onde a transição de cada conjunto ação-*build order* possui uma probabilidade.

3 Método

Neste trabalho foi utilizado Python 2.7 como a linguagem de programação para a implementação do método. Este motivo foi guiado principalmente pelo fato da biblioteca oficial de processamento de replays oferecida pela Blizzard Entertainment, Inc. (desenvolvedor do StarCraft II) ser em Python. A biblioteca de processamento de replays oficial, *s2protocol*(Blizzard Entertainment, Inc., 2013) é de código-fonte aberto e está disponível no GitHub. Demais códigos utilizados são parte de uma distribuição padrão do Python 2.7 ou foram desenvolvidos para o trabalho.

3.1 Extração dos dados

Os replays de StarCraft II são arquivos do tipo MoPAQ, um formato proprietário desenvolvido pela Blizzard Entertainment nos anos 90 para uso em seus jogos. Embora o formato seja proprietário é possível localizar na internet uma gama de implementações ou documentação desenvolvida por engenharia reversa. Um arquivo MPQ é análogo à um arquivo ZIP e contém um índice de arquivos e blocos de conteúdo. Nos *replays* há uma série de *stream* de eventos de jogo e neste trabalho somente será utilizado o stream chamado "Tracker Events"que são eventos cujo conteúdo é destinado para ferramentas que desejam processar os replays, ao contrário de informações úteis para a simulação do jogo. Este *stream* está armazenado como "replay.tracker.events"dentro do MPQ.

O *stream* de *Tracker Events* foi processado utilizando uma classe em Python que extrai as seguintes informações do evento:

- Nome (tipo) do evento;
- *Game loop* do evento (o número de iteração do loop principal do jogo) ;
- ID do jogador que gerou o evento;
- Estrutura de dados específica de cada evento;

explicar a conversão

Dentre os vários tipos de eventos contidos neste stream, 4 destes eventos são interessantes na análise:

NNet.Replay.Tracker.SPlayerStatsEvent : contém informações sobre o estado atual do jogador. De interesse na análise: suprimento em uso pelo exército e total de suprimento disponível.

NNet.Replay.Tracker.SUnitBornEvent evento disparado para cada unidade/estrutura criada no jogo. Este evento somente é despachado para unidades que aparecem no campo de batalha de forma "pronta", isto é, no momento em que a unidade pode ser vista pelo jogador, ela já pode ser controlada.

NNet.Replay.Tracker.SUnitInitEvent semelhante ao evento SUnitBornEvent, mas este evento é despachado para unidades que são construídas diretamente no campo de batalha e não estão disponíveis para jogo imediatamente no instante do evento. Embora a unidade/estrutura neste evento não seja imediatamente utilizável, no método proposto, apenas o tempo em que usuário inicia a construção é considerado.

NNet.Replay.Tracker.SUpgradeEvent evento disparado para cada melhoramento de exército realizado pelo jogador. Estes eventos são muito importantes pois podem indicar ou refinar a intenção do jogador.

Para o evento **SPlayerStatsEvent** são extraídas duas informações: a quantidade de suprimento utilizada pelo exército do jogador e o total de suprimento disponível para o jogador. Esta informação não é utilizada pelo método de classificação, mas é utilizada popularmente em sites de *build orders* de StarCraft II, uma vez que, durante o jogo, é mais simples indexar as ações baseado no suprimento usado e total do que no instante de tempo absoluto da ação.

Para os eventos **SUnitBornEvent**, **SUnitInitEvent** e **SUpgradeEvent** descritos anteriormente, apenas uma informação é extraída: o nome da unidade, estrutura ou melhoramento feito pelo jogador. Este nome é único para cada tipo de unidade, estrutura ou melhoramento.

Uma vez que após os 6 minutos de jogo, a partida se torna reativa, ao invés de algorítmica, a sequência de eventos é truncada até os 6 minutos de jogo. Isto garante que as informações tenham menor variabilidade.

explicar melhor isso

3.2 *Dataset* de referência

Como não há nenhum índice de replays e build-orders disponível publicamente o conjunto de replays utilizado para treinamento e validação foi classificado manualmente utilizando replays de diversos campeonatos do ano de 2016:

- Intel Extreme Masters 10
- Intel Extreme Masters 11
- WCS Spring Championship 2016

- WCS Summer Championship 2016
- DreamHack ZOWIE Open Valencia

quantidade de replays em cada

Para a classificação manual das estratégias cada replay foi assistido no jogo e classificado conforme as regras abaixo:

- A composição de exército de um jogador no instante do primeiro ataque realizado por ele;
- Se o jogador fosse atacado por outro e tivesse mais de 8 unidades perdidas, o *replay* era descartado;
- Caso não houvesse investida por parte dos dois jogadores até os 6 minutos de jogo, a composição no instante era classificada.

Um exército em StarCraft II pode ser composto por mais de 10 tipos de unidades diferentes, contudo isto é incomum. De forma geral, exércitos são compostos por um grande número de unidades básicas de ataque e algumas unidades de suporte. Na classificação, as unidades de suporte foram desprezadas na composição do exército. Esta regra é violada somente caso a unidade de suporte seja incomum ou trouxe um benefício significante para o jogador.

3.3 Treinamento

O treinamento foi realizado de forma simples: a sequência de ações para cada *replay* era iterada e o tempo de execução da ação era gravado em uma lista. Uma lista separada era usada para cada repetição. Por exemplo, supondo que as *build orders* BO_1 e BO_2 sejam duas amostras com o mesmo *label*:

Tabela 1 – Um exemplo de uma *build order* com as ações A_1 e A_2 ocorrendo nos tempos t_n.

BO_1		BO_2	
T	**A**	**T**	**A**
t_1	A_1	t_2	A_1
t_3	A_1	t_4	A_1
t_5	A_2	t_6	A_1
t_7	A_1	t_8	A_2

Dessa forma, as duas listas terão o seguinte conteúdo:

$$T_{A_1} = \{\{t_1, t_2\}, \{t_3, t_4\}, \{t_6, t_7\}\} \tag{3.1}$$

onde T_{A_1} é o vetor de tempos da execução de cada repetição da ação A_1 nos múltiplos *replays* processados. $\{t_1, t_2\}$ representa os tempos da primeira repetição, $\{t_3, t_4\}$ da segunda e $\{t_6, t_7\}$ da terceira.

$$T_{A_2} = \{\{t_5, t_8\}\} \tag{3.2}$$

onde T_{A_2} é o vetor de tempos da execução de cada repetição da ação A_2 nos múltiplos *replays* processados. Esta ação possui uma única repetição.

Seja $\mu(T_x, R)$ a média do vetor de tempo T_x para a repetição R e $\sigma(T_x, R)$ o desvio padrão do vetor de tempo T_x para a repetição R, então a distribuição p de probabilidade de uma ação pode ser definida como:

$$p_{T_x}(t, R) = e^{-\frac{(t-\mu(T_x,R))^2}{\sigma(T_x,R)^2}} \tag{3.3}$$

A Equação 3.3 apresenta a função distribuição de probabilidade para uma ação. Isto é, representa a probabilidade de uma ação qualquer A pertencer a uma *build order* BO_1 dado que a ação foi executada pelo jogador no instante t.

explicar porque esta função

Para o cálculo da média (μ) e desvio padrão (σ) os itens internos do vetor T_x são utilizados. Para o exemplo apresentado nas Equações 3.1 e 3.2, as médias são dadas pelos vetores das Equações 3.4 e 3.5, respectivamente:

$$\mu_{A_1} = \left\{\frac{t_1 + t_2}{2}, \frac{t_3 + t_4}{2}, \frac{t_6 + t_7}{2}\right\} \tag{3.4}$$

$$\mu_{A_1} = \left\{\frac{t_5 + t_8}{2}\right\} \tag{3.5}$$

O cálculo de desvio padrão para os vetores das Equações 3.1 e 3.2 foi omitido no exemplo pois a expressão é complexa e não influencia no entendimento e um valor genérico é apresentado nas Equações 3.6 e 3.7.

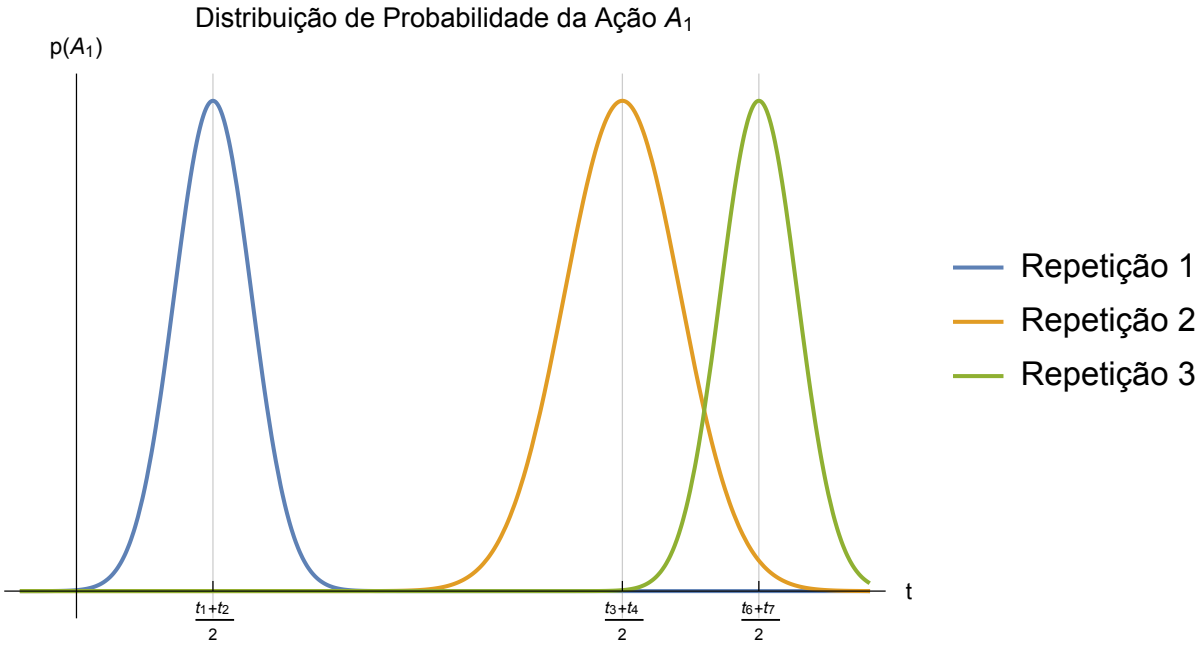
reescreve

$$\sigma_{A_1} = \{\sigma_{A_1,1}, \sigma_{A_1,2}, \sigma_{A_1,3}\} \tag{3.6}$$

$$\sigma_{A_1} = \{\sigma_{A_2,1}\} \tag{3.7}$$

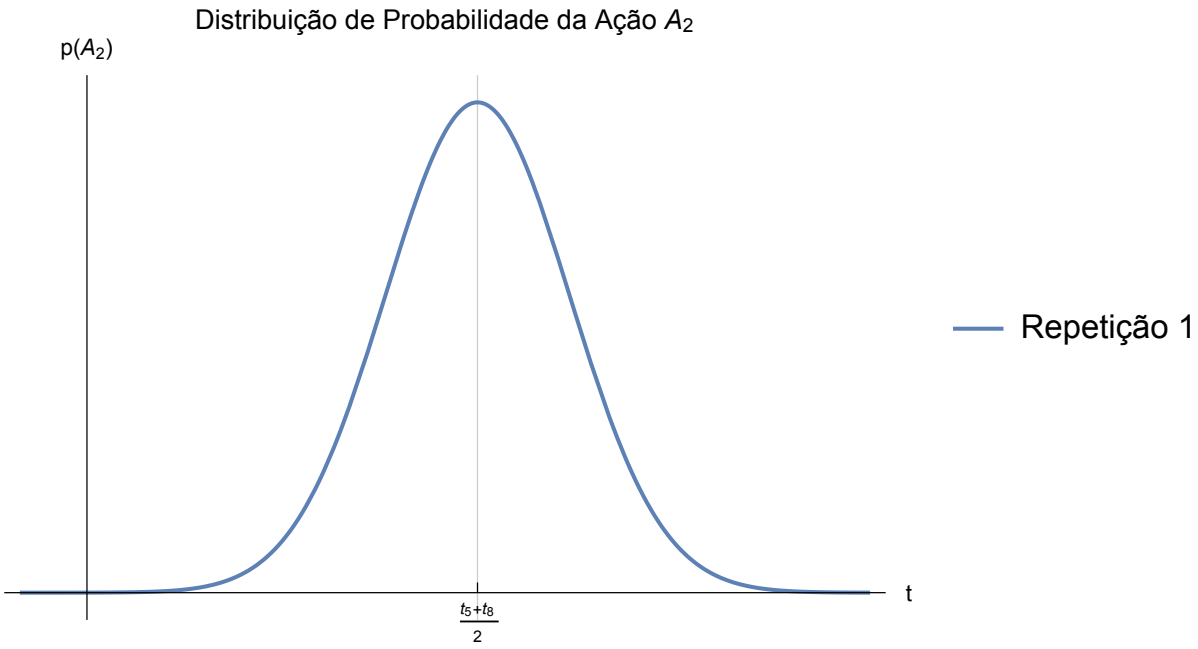
Com os resultados de média (Equações 3.4 e 3.5) e desvio padrão (Equações 3.6 e 3.7), é possível obter um gráfico do formato da distribuição de probabilidade conforme representado nas Figuras 2 e 3.

Figura 2 – Exemplo da distribuição de probabilidade da ação A_1. A curva em azul representa a primeira repetição da ação (nos tempos t_1 e t_2), a curva em laranja representa a segunda repetição (nos tempos t_3 e t_4) e a curva em verde representa a terceira repetição da ação (tempos t_6 e t_7).



3.4 Classificação

Figura 3 – Exemplo da distribuição de probabilidade da ação A_1. Como a ação A_2 somente é executada uma única vez no exemplo, apenas uma repetição é mostrada no gráfico. A ação corresponde aos instantes de tempo t_5 e t_8.



4 Resultados e Discussões

5 Conclusões

6 Propostas de Trabalhos Futuros

Referências Bibliográficas

Blizzard Entertainment, Inc. *s2protocol*. 2013. Disponível em: <https://github.com/Blizzard/s2protocol>.

RUGGERI, F.; KENETT, R. S.; FALTIN, F. W. *Encyclopedia of statistics in quality and reliability*. [S.l.]: Wiley Chichester, England, 2007.

SYNNAEVE, G.; BESSIERE, P. A bayesian model for opening prediction in rts games with application to starcraft. In: IEEE. *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. [S.l.], 2011. p. 281–288.

SYNNAEVE, G.; BESSIERE, P. A bayesian model for plan recognition in rts games applied to starcraft. *arXiv preprint arXiv:1111.3735*, 2011.

WEBER, B. G.; MATEAS, M. A data mining approach to strategy prediction. In: IEEE. *2009 IEEE Symposium on Computational Intelligence and Games*. [S.l.], 2009. p. 140–147.