Instituto Superior Técnico

MEEC

Programação Orientada a Objetos

Video Poker

Alunos:	*Número:*
Sérgio Neves	78111
David Carvalho	78469
Daniel Guerra	78885

14 de Maio de 2017

1 Introdução

Video Poker é um jogo de Poker virtual que foi bastante popular nos anos 80. O objectivo deste trabalho era implementar um variação deste jogo, o *Jacks or Better*, *Double Bonus 10/7*. Esta especifica variação do jogo tem uma taxa teórica de retorno de 100,2% , ou seja, teoricamente é possível ganhar dinheiro ao casino jogando com uma estratégia perfeita.

Foi necessário estudar a estrutura do código e a sua organização, isto porque foi importante pensar na extensibilidade do código a outras variantes e também a diferentes modos de jogo. Além disso, foi implementado o *advice* que simula a estratégia perfeita, com base do enunciado do projecto. Foi também implementado uma parte gráfica do programa (GUI).

2 Implementação

2.1 Cartas

Foi decidido usar uma classe para representar cada carta e uma classe para representar o baralho. Sempre que é necessário baralhar o baralho é criado um novo, uma vez que o *constructor* da respectiva classe inicializa o baralho como uma lista de cartas cuja posição de cada uma é aleatória.

Para inicializar o baralho mete-se uma carta de cada vez na lista que o representa numa posição aleatória entre as cartas já presentes na mesma.

2.2 Jogador

Usou-se uma classe para representar o jogador que contém a informação relativa ao mesmo, como os créditos, a presente mão de cartas, as estatísticas desde o início do jogo e o valor da última aposta.

A mão de cartas do jogador é também representada por uma classe que contém os métodos referentes à mesma que são necessários para a sua avaliação.

Para implementar a estratégia deste jogo é usada uma classe abstrata - *CardsTo* - que se divide em várias subclasses. Cada uma destas subclasses calcula o número de cartas para um certo de tipo de mão e/ou a existência de certos tipos de cartas na mão do jogador. Estas subclasses em conjunto com a avaliação da presente mão do jogador contém toda a informação necessária para informar o jogador de qual é o melhor conjunto de cartas a manter para maximizar o retorno de créditos.

- *CardsToHighCards* - Procura pelas diferentes *High Cards* e de seguida verifica a existência de combinações das mesmas pela ordem indicada na estratégia disponibilizada. É nesta classe que também se verifica a existência de 3 ou 4 cartas para um *royal flush*.
- *CardsToStraight* - Ordena a mão por ordem crescente do *rank* das cartas e como é necessário verificar se existem 3 ou 4 cartas para uma sequência as 3 primeiras cartas podem ser o possível princípio de uma sequência. Assim, é analisada uma destas 3 cartas de cada vez, procurando pelos 4 *ranks* seguintes na mão do jogador. Também é necessário verificar se a quarta carta da mão ordenada pode ser o início

de uma possível sequência para o caso das sequência com cartas altas, uma vez que o Ás é a primeira carta da mão ordenada (Exemplo: na mão ordenada $A47JK$, AJK são 3 cartas para uma sequência). Além disso também é contado o número de *high cards* presentes nas cartas que constituem a possível sequência e verifica-se se uma sequência é *outside* ou *inside* analisando se a carta que falta numa possível sequência de 4 cartas é constituída por cartas com *ranks* consecutivos ou não.

- *CardsToStraightFlush* - Esta classe segue o mesmo algoritmo usado na classe acima descrita, mas apenas verifica sequências do mesmo *suit* e além disso conta o número de cartas que faltam entre *ranks* consecutivos para formar a sequência. Isto é feito de modo a calcular o tipo do *straight flush draw*.
- *CardsToFlush* - Verifica o número de cartas cujo *suit* é o mais representado na mão do jogador e conta o número de *high cards* presentes nessas cartas.

2.3 Jogo

Para implementar todas as funções relativas ao jogo foi necessário pensar qual era a melhor forma de o manter extensível para novas vaiantes de jogo e investigar quais seriam os métodos comuns a cada variante.

Desta forma implementou-se o *Video Poker* como uma classe abstrata com métodos abstratos que dependem de cada variante de jogo e com métodos implementados que tratam da iteração do jogador com o jogo. Os métodos abstratos declarados são os que são necessários para que uma nova variante de jogo tenha as mesmas características do que foi pedido. Assumiu-se que as mãos que retornam créditos e os valores possíveis para apostas são diferentes para variantes distintas.

Como existem vários tipo de *Double Bonus* também é necessário dividir os métodos que são comuns a todos os tipos e os que são específicos a cada um. Assumiu-se que todos os tipos de *Double Bonus* têm em comum as mãos que retornam créditos, bem como os valores possíveis para apostar, no entanto como o valor de retorno de cada mão é diferente para cada tipo, a tabela de *rewards* é diferente e, consequentemente, o conselho para algumas mãos dado ao jogador também vai variar entre tipos de jogo.

3 Modos de Jogo

Nesta implementação do *Video Poker* existem 3 modos de o jogar: *Interactive, Debug e Simulation.*

3.1 Interactive Mode

Como este modo não tem nenhum limite de número de turnos, foi implementado um novo comando para se poder sair do jogo. Este comando é utilizado através da introdução do carater 'q'. este comando só pode ser utilizado depois de acabar um turno, ou seja, depois da mão do jogador ser avaliada e antes do jogador iniciar um novo turno, isto é, introduzir o comando de *deal*. A outra úncia maneira de o jogo terminar acontece caso o jogador fique sem créditos.

3.2 Debug Mode

Para testar aspetos específicos da implementação do jogo, este modo de jogo é usado. Para tal é necessário fornecer um ficheiro com a sequência de comandos a ser usados, bem como um ficheiro com uma sequência de cartas que representa o baralho. Para este efeito foram usados 5 pares de ficheiros:

- *card-file.txt* e *cmd-file.txt* - Com o objetivo de verificar os comandos básicos comparando o *output* com o que foi disponibilizado.
- *deckout-cards.txt* e *deckout-cmd.txt* - Com o objetivo de verificar o comportamento do programa quando o baralho não tem cartas suficientes para dar ao jogador.
- *some-plays-cards.txt* e *some-plays-cmd.txt* - Com o objetivo de verificar a função de *advise* para algumas das *difficult hands* disponibilizadas.
- *states-cards.txt* e *states-cmd.txt* - Com o objetivo de testar a possibilidade ou impossibilidade de usar determinados comandos em certos estados do jogo.
- *win-hands-cards.txt* e *win-hands-cmd.txt* - Com o objetivo de verificar se todas as possíveis mãos que retornam créditos são bem avaliadas.

3.3 Simulation mode

Este modo de jogo serve principalmente para calcular o retorno médio do jogo. Como a estratégia disponibilizada não era a perfeita, mas sim uma mais simples, o retorno esperado não é o de $100,2\%$. Deste modo para avaliar o retorno médio desta variante de jogo com a estratégia implementada foi calculada a médio do retorno para diferentes número de mãos jogadas, com um crédito inicial de 10000, para vários valores de aposta. Foram feitas 10 execuções para calcular a média. Os resultados são apresentados na seguinte tabela.

Tabela 1: Retorno Médio (%)

	100 deals	1000 deals	10000 deals	100000 deals	1000000 deals
1 credit	100,10	99,44	97,42	89,91	15,15
2 credit	99,71	100,36	94,36	77,04	0,00
3 credit	99,76	97,62	99,98	74,25	0,00
4 credit	100,70	99,51	95,77	99,61	9,21
5 credit	100,42	99,28	101,62	109,51	63,46

4 Graphic User Interface(GUI)

Foi implementada uma interface gráfica para ser mais conveniente e fácil o utilizador conseguir jogar. As escolhas para este modo, para além de estéticas, foram também a pensar nas necessidades do jogador. Os modos *simulation* e *de-bug* não pareceram cruciais para o jogador, o que resultou na sua não utilização. A escolha de botões para a representação das cartas pareceu obvia sendo que foi decidido que as cartas a serem mantidas seriam decididas pelo pressionamento das mesmas. O mesmo raciocínio foi aplicado ao botão de *deal*, sendo mais tarde definido também como o botão que representa o *deck*, este no

entanto poderia não parecer obvio, portanto foi implementada uma *tool tip* de modo a que fosse possível fornecer esta informação ao jogador. De modo a que o campo de jogo não estivesse muito obstruído, as estatísticas foram colocadas numa janela a parte, de modo a que, caso o jogador não as quisesse visíveis as podia esconder. Por fim foram colocadas várias mensagens *pop up*, de modo a informar o jogador caso ele estivesse a fazer algo de maneira incorreta.

5 Conclusão

Como se pode verificar no modo *simulation*, com um número elevado de iterações, a percentagem de retorno tende para 0, sendo que este é um jogo de sorte, não desprova a estratégia utilizada, no entanto demonstra que esta não é perfeita.

Sendo o objectivo inicial do trabalho simular uma variação do jogo Video Poker, pode-se dizer que este foi cumprido com sucesso. No entanto existem algumas coisas que podiam ter sido melhoradas, como por exemplo no GUI podia-se pôr a janela de jogo *resizable*, e dar uma escolha ao jogador da parte traseira das cartas, tudo isto são possibilidades para uma futura melhoria do projeto.