GRH – Genetic Rush Hour: Uma heuristica genética na solução de problemas Pspace-completo

Ana J. Fontoura[1], Tauame A. Pacce[1], Jacques N.C. Schreiber[1]

[1] Departamento de Informática, Bacharelado em Ciência da Computação, UNISC – Universidade de Santa Cruz do Sul, Santa Cruz do Sul, RS, Brasil

{ana.fontoura, tauamepacce}@hotmail.com, jacques@unisc.br

Resumo. Este artigo descreve uma solução para resolver o quebra-cabeça Rush Hour utilizando um algoritmo genético. Este quebra-cabeça é um problema do tipo Pspace-completo. Na teoria da complexidade computacional, PSPACE é o conjunto de todos os problemas de decisão que podem ser resolvidos por uma máquina de Turing usando uma quantidade polinomial de espaço, e é dito PSPACE-completo se pertence à classe de complexidade PSPACE e todos os problemas em PSPACE podem ser reduzidos a ele em tempo polinomial. A heurística implantada para solucionar o problema individualiza os indivíduos da população do algoritmo genético a partir dos movimentos possíveis no quebra-cabeça. A heurística será discutida em detalhes no que diz respeito ao seu efeito sobre a população, cálculo de aptidão, mutação e operadores de crossover.

Palavras chave: Rush Hour puzzle, Algoritmos Genéticos, Heurísticas.

1 Introdução

Os jogos do tipo quebra-cabeças têm recebido muita atenção da comunidade de Inteligência Artificial nos últimos anos [1, 3]. No entanto, alguns quebra-cabeças mais complexos foram relativamente negligenciados pelos pesquisadores, em [2] há uma revisão completa sobre o tema. O jogo Rush Hour é um desses jogos difíceis e foi considerado assim por ser um problema PSPACE-completo (ou seja, mais difícil do que problemas NP-completos, se NP $\subset$ PSPACE). A versão comercial do jogo é jogada em uma matriz 6x6, simulando um estacionamento repleto de carros e caminhões. O objetivo é encontrar uma sequência de movimentos legais para os veículos de modo a liberar a passagem do veículo vermelho em direção à saída. (ver Fig. 1). (Nota: Ao longo deste trabalho, usaremos o termo “carro” quando nos referirmos a um elemento que ocupa dois espaços, “caminhão” quando se refere a um elemento que ocupa três espaços, e "veículo" ao se referir a qualquer um).

Uma abordagem na resolução de problemas importantes no campo da inteligência artificial é o da busca heurística. Um dos algoritmos de busca heurística mais importantes é o aprofundamento iterativo A* (IDA*) [4, 5], com várias versões aprimoradas, incluindo matrizes de transposição [6, 7], movimentos ordenados [8], base de dados com padrões de jogos [9].

Fig. 1. Layout típico de um tabuleiro de Rush Hour.

Este método é largamente utilizado para resolver jogos do tipo quebra-cabeça (por exemplo, [11, 10]). A ideia em que o IDA* e algoritmos similares se baseiam está na noção de fazer o movimento que mais se aproxima de um estado objetivo. Tais aproximações são encontradas, por meio de uma função heurística. Ao aplicar a função heurística para estados alcançáveis a partir dos estados atuais considerados, torna-se possível selecionar as alternativas mais promissoras no início do processo de pesquisa, possivelmente reduzindo a quantidade de esforço de pesquisa (normalmente medido em número de nós expandidos) necessários para resolver um determinado problema. A redução do esforço computacional está fortemente associada à qualidade da função heurística utilizada: empregando uma função perfeita significa simplesmente "navegar" para a solução (ou seja, nenhuma pesquisa de fato ocorreria, ou nenhum movimento inútil seria realizado), enquanto que, mesmo utilizando uma função ruim podemos tornar a busca mais eficiente do que uma busca cega, como a busca em largura (BFS) ou em profundidade (DFS). Até o momento, não há heurísticas eficientes relatadas para a solução do quebra-cabeça Rush Hour.

Este artigo está dividido em cinco (5) seções: após a apresentação do tema da nossa pesquisa na introdução a seção dois explica o jogo Rush Hour e as suas regras. A seção três detalha os aspectos mais importantes do algoritmo genético implementado com destaque à função de fitness e regras para efetuar o crossover. A seção quatro valida o desempenho do nosso GRH comparando-o com os recordes atuais. Na última seção apresentamos as nossas considerações.

2 O jogo Rush Hour

O jogo Rush Hour desenvolve-se num tabuleiro de dimensão 6 × 6, num total de 36 posições quadriculares, delimitado por um conjunto de paredes externas e com uma única saída. Para simplificar, a saída encontra-se sempre na mesma posição, ou seja, do lado direito do tabuleiro, na 3ª linha do mesmo. Dentro do tabuleiro estão

colocados diversos veículos entre carros e caminhões de comprimento correspondente a 2 ou 3 quadrículas, dispostos horizontal ou verticalmente. A disposição de cada veículo define univocamente o tipo de movimentos que lhe é permitido efetuarem. Um veículo disposto horizontalmente, apenas lhe é permitido deslocar-se horizontalmente ao longo da linha em que está colocado. Da mesma forma, um veículo disposto verticalmente, apenas lhe é permitido deslocar-se verticalmente ao longo da coluna em que está colocado.

O objetivo do jogo é determinar uma sequência de movimentos necessários para os vários veículos, de tal forma a permitir abrir caminho para que o veículo do jogador, identificado pela cor vermelha (veja Fig. 1), possa sair pela única saída disponível. Além das restrições óbvias, tais como o fato de que todos os movimentos dos veículos terem de ser efetuados dentro do próprio tabuleiro, existem outras restrições igualmente simples:

1. Qualquer veículo apenas pode se deslocar na direção correspondente a sua disposição inicial (ou seja, na linha ou coluna onde está posicionado);
2. O movimento de qualquer veículo nunca pode colidir com o posicionamento de outro veículo (ou seja, os veículos não podem se sobrepor);
3. Qualquer movimento, em qualquer sentido, permite apenas deslocar um veículo em apenas uma posição (ou célula ou quadrado) de cada vez;

A primeira restrição decorre diretamente da descrição anteriormente feita do posicionamento dos veículos. A segunda, referida anteriormente, prende-se ao fato de certos veículos poderem estar imobilizados, restringindo de forma óbvia os movimentos que cada veículo pode fazer num dado instante. Finalmente a terceira restrição é imposta apenas para efeitos de contabilização do esforço despendido (número de movimentos utilizados na procura da solução).

A heurística proposta neste artigo utiliza um algoritmo genético que deve assim ser capaz de, dada uma configuração do tabuleiro com indicação do posicionamento de todos os veículos, gerar a sequência de movimentos necessários para retirar o veículo vermelho através da saída indicada. Todos os movimentos gerados devem ser movimentos válidos a luz das regras do jogo e de acordo com a configuração do tabuleiro. Dado tratar-se de um jogo, o número de movimentos indicados deve ser o menor possível para conseguir o objetivo pretendido. A contabilização do número de movimentos efetuados pelo jogador no sentido de abrir caminho para o seu veículo sair do engarrafamento, define a sua pontuação no jogo. Obviamente quanto menor for a pontuação, menor terá sido o esforço dispendido e melhor a classificação da heurística. Numa competição, dadas duas sequências de movimentos possíveis como solução para o problema, obtidas por dois jogadores distintos, será considerado vencedor o jogador que produzir a sequência que tiver menos movimentos.

Dado tratar-se de uma resolução computacional do problema, pretende-se que a obtenção da sequência final seja feita com o menor dispêndio de recursos possível.

3 GRH – implementação do algoritmo genético

O algoritmo GRH será descrito nesta seção, e para tanto será usado o ciclo básico de execução de um AG para explicar cada componente e cada processo no ciclo de execução do GRH. Um algoritmo genético básico possui as seguintes etapas:

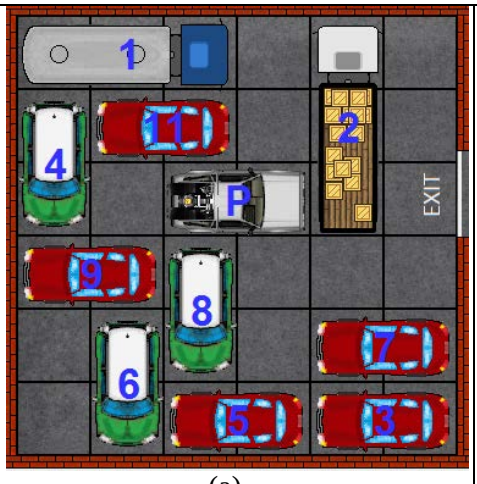
a. Geração da população inicial
b. Cálculo da aptidão de cada indivíduo
c. Verificação da convergência, é a solução?
    a. Sim? Encerra o processo
    b. Não? Seleciona os pares mais aptos da população
d. Procede-se a reprodução (crossover)
e. Procede-se a mutação se for o caso
f. Gera-se a nova população e retorna-se ao segundo passo (letra b).

3.1 Geração da População Inicial

A população é composta por um número limitado de indivíduos representando o problema, portanto antes de gerar a população inicial será explicado como cada indivíduo representa o ambiente Rush Hour.

3.1.1 O Cromossomo no GRH

Cada indivíduo, ou cromossomo é representado através de uma matriz de 6x6 de números inteiros, onde cada quadrícula ou célula desta matriz representa uma posição no tabuleiro e pode conter os seguintes valores: ‘0’ para um espaço vazio, um número par para espaços ocupados por carros na vertical, um número impar para espaços ocupados por carros na horizontal e a constante no valor 51 para o espaço ocupado pelo carro do jogador. Veja a Fig. 2:



(a)

1	1	1	0	2	0
4	1	1	0	2	0
4	0	51	51	2	0
9	9	8	0	0	0
0	6	8	0	7	7
0	6	5	5	3	3

(b)

Fig. 2. Exemplo de representação do cromossomo.

Na Fig. 2b, é possível verificar a representação do tabuleiro apresentado na Fig. 2a. Pode-se, por exemplo, observar que o caminhão cinza disposto no canto superior esquerdo está mapeado através do número ímpar ‘1’ disposto três vezes na horizontal. A cada novo veículo disposto na horizontal atribui-se um novo número ímpar, assim como a cada novo veículo disposto verticalmente atribui-se um novo número par. O veículo cinza (letra P na Fig. 2a) representa o veículo do jogador e foi representado pelo número 51 no tabuleiro da Fig. 2b.

3.1.2 Geração da População Inicial

A população inicial não é gerada aleatoriamente. O Rush Hour é um jogo de tabuleiro, onde o jogador é desafiado a resolver um problema apresentado. Da mesma forma, o GRH recebe como dado de entrada um tabuleiro que deve ser resolvido, então, a partir deste é criada uma população inicial formada por todos os possíveis tabuleiros gerados com apenas um movimento. A seguir para cada um destes tabuleiros gerados, é repetido o processo de criar todos os tabuleiros possíveis com um novo movimento. Ao final deste processo, tem-se a população inicial, que é formada basicamente por todos os tabuleiros possíveis com dois movimentos a partir do inicial. Portanto, se a solução estiver há dois movimentos de distância o problema já é resolvido pela simples geração da população inicial. A Fig. 3 mostra alguns indivíduos da população inicial gerada a partir do tabuleiro apresentado na Fig. 2a.

Fig. 3. Indivíduos da População Inicial.

3.2 Cálculo da aptidão de cada indivíduo – Fitness

Cada indivíduo gerado é avaliado de acordo com uma função que atribui um valor numérico. Dessa forma será possível avaliar a aptidão de todos os componentes de uma população. O fitness é calculado em três etapas, cada etapa adotou uma heurística distinta:

Etapa 1. A primeira heurística consiste em localizar cada veículo que está bloqueando o caminho de saída do carro cinza (carro do jogador). Recursivamente, para cada veículo bloqueador verifica-se a existência de outros veículos bloqueadores. Esse processo se repete até que não se encontrem mais veículos bloqueando o caminho sendo analisado. A métrica utilizada no cálculo desta heurística atribui um valor numérico para cada veículo. A função retorna valores maiores para os veículos que estão bloqueando diretamente o carro do jogador e valores proporcionalmente menores para os veículos que não estão bloqueando diretamente o carro cinza. O valor desta heurística inicia com um máximo teórico, e então subtrai-se cada um dos valores dos carros. Conforme a Fig. 4, os caminhões (números 6 e 2) retornariam valores maiores que o carro de número 1 pois os caminhões estão bloqueando diretamente o caminho de saída do carro cinza. É através desta heurística que é verificado se o tabuleiro é uma solução para o problema, pois se a soma dos valores numéricos dos veículos bloqueadores for zero significa que não há mais bloqueio no caminho de saída. Um cromossomo com muitos veículos bloqueadores (direta e/ou indiretamente) atingirá um soma maior do que um cromossomo com uma quantidade menor de veículos bloqueadores, ou seja, quanto menos veículos bloqueadores melhor será a aptidão deste cromossomo.

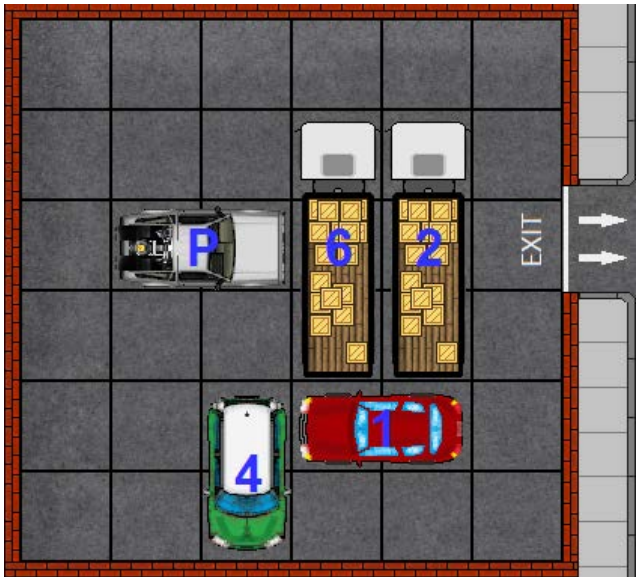


Fig. 4. Cenário hipotético.

Etapa 2. A segunda heurística consiste em somar o número de movimentos possíveis para cada veículo do tabuleiro.

Etapa 3. A terceira e última heurística conta o número de vezes que um tabuleiro se repetiu. Conforme o algoritmo genético vai executando, este pode

eventualmente gerar tabuleiros que já foram criados anteriormente. Esta repetição não deve ser evitada, pois cada vez que um tabuleiro é criado novamente, ele tem a possibilidade de gerar filhos que não foram gerados anteriormente. Porém, as chances de serem gerados filhos inéditos diminuem a cada vez que este tabuleiro é gerado novamente.

O valor final retornado pelo fitness, é o resultado da seguinte equação (cada heurística possui pesos diferentes):

$$\text{Fitness} = (\text{Etapa1} * 0{,}8) + (\text{Etapa2} * 0{,}2) - (\text{Etapa3} * 5)\ . \qquad (\mathbf{1})$$

3.3 Seleção dos mais Aptos – Roleta

O GRH possui uma função de roleta que recebe uma lista de tabuleiros gerados como parâmetro e retorna uma lista dos selecionados. Para selecionar os tabuleiros, a função de roleta pega temporariamente o valor numérico do fitness de cada um dos tabuleiros e subtrai deste o valor do pior fitness dividido por 2. Optou-se por fazer esta subtração para que o desvio padrão dos valores de fitness aumente. Esse procedimento permitiu uma seleção mais satisfatória dos indivíduos mais aptos.

3.4 Crossover

A função de crossover recebe como parâmetro a lista dos indivíduos selecionados pela roleta, agrupa estes em casais e para cada casal é executado o cruzamento ou crossover. Este procedimento ocorre da seguinte maneira:

a. Para cada pai, cria-se uma lista (lista paiA e lista paiB) de todos os tabuleiros possíveis a partir deste. Entende-se como tabuleiro possível aquele tabuleiro passível de ser alcançado a partir de um movimento no respectivo tabuleiro pai.
b. Cada um dos tabuleiros possíveis gerados é avaliado e recebe uma nota pela função de fitness.
c. Aplica-se a roleta para cada uma das duas listas (lista paiA e lista paiB).
d. Após proceder com a roleta formam-se pares de tabuleiros (1 elemento gerado a partir da lista paiA, e outro tabuleiro gerado a partir da lista paiB).
e. Para cada par formado: Analisam-se os carros que ambos os tabuleiros possuem na mesma posição, estes serão replicados nos tabuleiros filhos gerados. Então para cada carro encontrado em posições diferentes nos dois tabuleiros, faz-se o seguinte:
    a. Se o carro encontrado no pai1, pode ir para a posição do mesmo carro no pai2, então executa este movimento e este será um dos filhos gerados.
    b. Se o carro encontrado no pai2, pode ir para a posição do mesmo carro no pai1, então executa este movimento e este será um dos filhos gerados.

Este processo se repete até que o número de filhos gerados se aproxime ao máximo do configurado na interface do programa. A motivação para este complexo método de crossover deveu-se à necessidade da geração de movimentos inéditos.

3.5 Mutação

A função de mutação executa uma iteração (loop) que percorre todos os indivíduos de uma geração. Para cada indivíduo é atribuído um valor aleatório entre 0 e 100, se esse valor atingir 1, o indivíduo será mutado. A mutação consiste em escolher um carro aleatoriamente no tabuleiro e executar um movimento aleatório com ele. Após executada a mutação, o tabuleiro gerado recebe um novo valor de fitness atualizado.

4 Validação

A validação da eficácia do GRH foi feita comparando o desempenho do nosso software com o melhor desempenho encontrado na literatura. O desempenho é medido pelo número de movimentos que possibilitem o carro sair do estacionamento, a menor quantidade de movimentos representa o recorde para o respectivo tabuleiro.

Esses recordes estão apresentados no site http://www.thinkfun.com/playonline. Neste site existe uma versão online do Rush Hour para ser jogado por um ser humano com quatro níveis de dificuldade: Fácil, Médio, Difícil e Especialista e para cada nível, 25 configurações possíveis de tabuleiros, totalizando cem desafios distintos. Cada tabuleiro representa um desafio de crescente dificuldade com o respectivo número mínimo de movimentos, ou seja, cada tabuleiro possui um recorde associado.

Com o objetivo de avaliar o GRH foram selecionados quarenta tabuleiros, os dez primeiros desafios de cada nível (Fácil, Médio, Difícil e Especialista). Os resultados dessa avaliação estão apresentados na Tabela 1. A tabela está organizada em seis (6) colunas, a primeira indicando o nível do jogo, a segunda apresenta a sequencia dentre as dez (10) rodadas dentro de cada nível. As informações mais relevantes são apresentadas na terceira e quarta coluna que apresentam o recorde definido no site e o obtido pelo nosso GRH. As duas últimas colunas apresentam a média de movimentos nas cinco (5) tentativas efetuadas pelo GRH para cada rodada e o tempo médio de execução.

O resultado da análise da tabela 1 demonstra que em trinta e cinco por cento dos casos (35%) o GRH atinge o recorde definido para cada rodada. Devemos observar que o melhor desempenho ocorreu no nível “fácil” onde o recorde foi atingido em oitenta por cento dos casos (80%). À medida que a complexidade dos tabuleiros aumenta a eficácia do GRH diminui, entretanto isso nos estimula a aprimorar a nossa heurística já vislumbrando melhor desempenho.

Um resultado a ser destacado é o tempo médio para o GRH solucionar os quarenta tabuleiros propostos. O tempo foi inferior a meio segundo em setenta e sete por cento dos casos (77,5% para ser exato). Esse valor demonstra que a heurística pode se tornar mais complexa, em detrimento do tempo de execução, possivelmente aprimorando o desempenho do GRH e ainda com um tempo aceitável de execução.

Tabela 1. Resultados obtidos, em número de movimentos, após 5 execuções do GRH para cada jogo.

Nível	Jogo	Mínimo Possível	Mín. GRH	Média GRH	Tempo Médio GRH (s)
Fácil	1	13	13	14	0.008
Fácil	2	11	11	11.8	0.0336
Fácil	3	28	30	31.2	0.0658
Fácil	4	12	12	13.2	0.0102
Fácil	5	13	13	14.8	0.0026
Fácil	6	21	21	21.4	0.0102
Fácil	7	24	24	25.6	0.0404
Fácil	8	18	18	18.8	0.0444
Fácil	9	21	23	23.4	0.0518
Fácil	10	19	19	19	0.0374
Médio	1	33	33	33.8	0.131
Médio	2	28	30	34.4	0.199
Médio	3	31	34	37.8	0.1954
Médio	4	34	34	37.2	0.1118
Médio	5	33	37	39.2	0.317
Médio	6	39	45	49	0.6448
Médio	7	28	32	33.2	0.2038
Médio	8	32	33	36.6	0.207
Médio	9	34	35	37.4	0.0754
Médio	10	28	28	29.2	0.0428
Difícil	1	35	38	40	0.1724
Difícil	2	38	42	45	0.2852
Difícil	3	42	42	43.2	0.1234
Difícil	4	37	40	42.6	0.237
Difícil	5	38	41	43.6	0.175
Difícil	6	42	50	54.2	0.4476
Difícil	7	40	41	46.2	0.1342
Difícil	8	35	38	40	0.2486
Difícil	9	36	36	39.2	0.4474
Difícil	10	31	34	35.8	0.1498
Especialista	1	79	79	80.4	0.7196
Especialista	2	93	97	103.6	1.3816
Especialista	3	86	93	96.6	1.0534
Especialista	4	62	66	69.2	0.4582
Especialista	5	85	89	90.6	0.5236
Especialista	6	66	71	73.8	0.631
Especialista	7	73	75	80.8	0.6444
Especialista	8	75	78	81.2	0.5402
Especialista	9	66	80	81.6	0.7614
Especialista	10	73	79	81.4	0.434

5 Conclusões

Em primeiro lugar, é de se notar que não existem ainda muitos trabalhos que utilizam algoritmos genéticos na resolução do Rush Hour. Vários estudos existem, porém utilizando outros algoritmos conforme observado na introdução deste artigo. O algoritmo genético desenvolvido para a resolução do Rush Hour utilizou uma estratégia diferenciada para o cruzamento e os resultados obtidos na validação demonstram que há muito trabalho a ser desenvolvido, mas que estamos na direção certa. Esta conclusão é devido aos trinta e cinco pontos percentuais em que o desempenho do nosso GRH foi igual aos recordes obtido até o momento.

Os resultados obtidos em relação ao tempo computacional foram satisfatórios, apresentando tempo de processamento bastante reduzido para a maioria das instâncias testadas.

Destaca-se, em especial, é que o procedimento heurístico proposto foi capaz de resolver todos os tabuleiros, independentemente do nível de complexidade. A heurística proposta mostrou-se adequada para aplicações reais, pois é capaz de produzir boas soluções em tempo computacional reduzido.

Referências

1. Hearn, R. A.: Games, puzzles, and computation. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science (2006).
2. Kendall, G., Parkes, A., Spoerer, K.: A survey of NP-complete puzzles. In: International Computer Games Association Journal (ICGA), vol. 31, pp. 13--34 (2008).
3. Robertson, E., Munro, I.: NP-completeness, puzzles and games. In: Utilas Mathematica, vol. 13, pp. 99--116 (1978).
4. Hart, P. E., Nilsson, N. J., Raphael, B.: A formal basis for heuristic determination of minimum path cost. In: IEEE Transactions on Systems Science and Cybernetics, vol. SSC 4, pp. 100 (1968).
5. Korf, R. E.: Depth-first iterative-deepening: An optimal admissible tree search. In: Artificial Intelligence, vol. 27(1), pp. 97--109 (1985).
6. Frey, P. W.: Chess Skill in Man and Machine. Springer-Verlag New York, Secaucus (1979).
7. Taylor, L. A., Korf, R. E.: Pruning duplicate nodes in depth-first search. In: AAAI-93 Proceedings, pp. 756--761 (1993).
8. Reinefeld, A., Marsland, T. A.: Enhanced iterative-deepening search. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16(7), pp. 701--710. IEEE Computer Society, Los Alamitos (1994).
9. Felner, A., Korf, R. E., Hanan, S.: Additive pattern database heuristics. In: Journal of Artificial Intelligence Research, vol. 22, pp. 279--318 (2004).
10. Korf, R. E.: Finding optimal solutions to rubik's cube using pattern databases. In: AAAI/IAAI, pp. 700--705 (1997).
11. Junghanns, A., Schaeffer, J.: Sokoban: A challenging single-agent search problem. In: IJCAI Workshop on Using Games as an Experimental Testbed for AI Research, pp. 27--36. Universiteit (1997).