Universidade do Porto

Faculdade de Engenharia

FEUP

Pesquisa aplicada à evacuação

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Hugo Vaz Neves - up201104178 - up201104178@fe.up.pt

Rúben José da Silva Torres - up201405612 - up201405612@fe.up.pt

Diogo Henrique de Almeida Silva Pereira - up201505318 - up201505318@fe.up.pt

8/4/2018

Índice

Objetivo

O tema do projeto que se encontra em desenvolvimento, no âmbito da cadeira de Inteligência Artificial, é o de pesquisa aplicada à evacuação de um conjunto de turistas retidos numa montanha.

O objetivo do grupo, passa por determinar o percurso ótimo para evacuar todos os turistas que se encontram em locais distintos (versão 2 do problema) ou no mesmo local(versão 1) no menor tempo possível, sendo que, para tal, estão disponíveis veículos de transporte com capacidade limitada que se encontram localizados em locais específicos.

Os locais de "pick up" serão vistos como "nodes" para o nosso algoritmo, e podem variar em número.

Especificação

Especificação do problema

Para representar um o número de turistas num determinado local, foi criado o predicado local(Coordenada X, Coordenada Y, Número de pessoas). Este predicado é utilizado para representar o número de turistas retidos numa montanha.

Além deste, para representar um ponto estratégico, é utilizado o predicado ponto_estrategico(ponto(Coordenada X, Coordenada Y, Número de pessoas)).

Estados

Na primeira iteração do nosso projecto, a quantidade de pessoas no local de abrigo era considerada para o problema de pesquisa um estado, sendo o estado inicial o local de abrigo vazio, e o final o local de abrigo com todos os turistas evacuados.

Na nova iteração, usamos estados iniciais e estados finais. Um estado é composto por Locais e Pontos.

Um Local é composto por coordenadas X e Y e uma variável que armazena a quantidade de pessoas ainda no local para evacuar.

Um Ponto é o conjunto de coordenadas X e Y e o número de pessoas no veículo nesse ponto.

Resumindo, um estado é uma lista composta por duas outras listas, e pode ser representado da seguinte maneira:

- Estado =[Locais,Pontos];
- Locais=[local(X,Y,Quantidade de pessoas a evacuar),...];
- Pontos= [ponto(X,Y,Pessoas no veículo),...]

Funções de transição

Para o problema de pesquisa existem três funções de transição:

- A primeira função de transição verifica se existem turistas retidos na montanha suficientes para encher o veículo, sendo verdade evacua-os imediatamente para o local de abrigo.
- A segunda função de transição verifica se o número de turistas retidos na montanha não é suficiente para encher o veículo, nesse caso, os turistas entram no veículo mas este mantém-se na montanha caso seja necessário a utilização do mesmo para evacuar pessoas noutra montanha.
- A terceira função de transição evacua veículos (com pelo menos 1 turista) que apesar de terem espaço para evacuar mais pessoas, não vão ser necessários para evacuar turistas noutras montanhas.

Todas as funções de transição têm como objetivo devolver o estado das montanhas e o estado dos veículos e o custo da utilização de um veículo para evacuação de turistas.

Heurísticas

A função heurística utilizada para este problema de pesquisa tenta prever o custo necessário para evacuar os restantes turistas retidos. Para o cálculo da função heurística é feita a média da distância entre todos os veículos e todos as montanhas com turistas por evacuar, sendo multiplicada esta média pelo número de viagens que se prevê serem necessárias efetuar.

Algoritmos de pesquisa a aplicar

Tendo em conta a representação do problema, o grupo acha apenas necessário utilizar um algoritmo tipo A* para resolver o problema de pesquisa.

Para o caso em concreto este algoritmo a cada iteração procura entre todas as transições possíveis a melhor em relação ao estado final , guardando para cada nó as transições utilizadas para chegar ao mesmo e o custo associado.

Ou seja, foi feita uma implementação clássica do algoritmo sem muitas alterações, no entanto há uma grande diferença que o torna extremamente mais rápido em alguns casos. A condição de paragem do algoritmo A* clássico é feita quando o estado que estamos avaliar pode ser considerado como estado final.

No algoritmo implementado além desta condição de paragem temos outra que acontece quando já não existem mais veículos em pontos estratégicos diferentes ou com menor custo de viagem que o local final. As viagens a partir deste momento serão sempre feitas entre o local final e as montanhas com pessoas por evacuar, sendo então o resto da resposta óbvio e podemos então desta maneira considerar que o algoritmo já encontrou uma boa solução.

Desenvolvimento

Para o desenvolvimento da aplicação foi apenas utilizado a linguagem prolog sem recurso a api's ou ferramentas extra. A aplicação foi feita com recurso ao Sicstus prolog.

A estrutura da aplicação pode ser dividida em 3 ficheiros do tipo ".pl":

- **algorithms.pl:** este contém a implementação das funções de transição, da função heurística e claro do algoritmo implementado (A*), além disso também contém funções utilitárias como average(List, Average) (calcula a média de uma lista) e replace(Index, List , Element, NewList)(troca um elemento especificado da lista por outro dado).

- **problem.pl:** Este ficheiro vai conter o nosso problema, para o programa funcionar este apenas pode conter os seguintes predicados:
    - local_inicial(local(coordenada X, coordenada Z, número de pessoas)) - Este predicado representa uma montanha com a sua localização e número de turistas retidos, podemos ter qualquer quantidade de montanhas.
    - local_final(X,Y) - Apenas pode existir 1 e 1 só local_final, este predicado representa um local de abrigo para onde as pessoas serão evacuadas.
    - ponto_estrategico(ponto(Coordenada X, Coordenada Y, 0)) - Este predicado representa um ponto estratégico ou seja um veículo.
    - capacidade_veiculos(Capacidade) - Apenas pode existir apenas 1 e um só predicado capacidade_veiculos, este representa a quantidade máxima de turistas que um veículo pode evacuar ao mesmo tempo.
- **solver.pl:** este ficheiro contém as funções necessárias para correr e testar o programa (solve_problem e runtime(X)), devemos alterar este ficheiro caso desejarmos utilizar outro problema incluido num ficheiro com nome diferente do "problem.pl".

Experiências

Tendo em conta que a função heurística é admissível, é esperado com aplicação que o algoritmo A* devolva uma solução ótima num curto espaço de tempo.

De modo a validar o desempenho do programa foram realizadas várias experiências, com diferentes dimensões do nosso problema, foram medidos os resultados com o objetivo de comparar tanto a qualidade das soluções obtidas para cada algoritmo como os recursos necessários para calcular cada solução

Para fazer estes testes basta correr o predicado "runtime(X).", com esta informação é possível validar a solução do algoritmo A*.

Realizamos três principais experiências:

1) Alteração no número de pontos (nós)
2) Alteração das distâncias totais;
3) Alteração da capacidade de carga;

1) Numero de pontos:

Usamos duas amostras diferentes, uma linear e uma aleatória;

Amostra linear:

```
%locais onde existem pessoas para evacuar

local_inicial(local(0,1,30)).

local_inicial(local(1,2,30)).
local_final(10,1).

capacidade_veiculos(5).

%ponto estratégicos
%local(Coordenada X, Coordenada Y, número de pessoas na carrinha)
ponto_estrategico(ponto(1,1,0)).
ponto_estrategico(ponto(2,2,0)).
ponto_estrategico(ponto(4,4,0)).
ponto_estrategico(ponto(3,3,0)).
ponto_estrategico(ponto(5,5,0)).
ponto_estrategico(ponto(6,6,0)).
ponto_estrategico(ponto(7,7,0)).
ponto_estrategico(ponto(8,8,0)).
ponto_estrategico(ponto(9,9,0)).
```

Resultados:

Amostra aleatoria:

%locais onde existem pessoas para evacuar

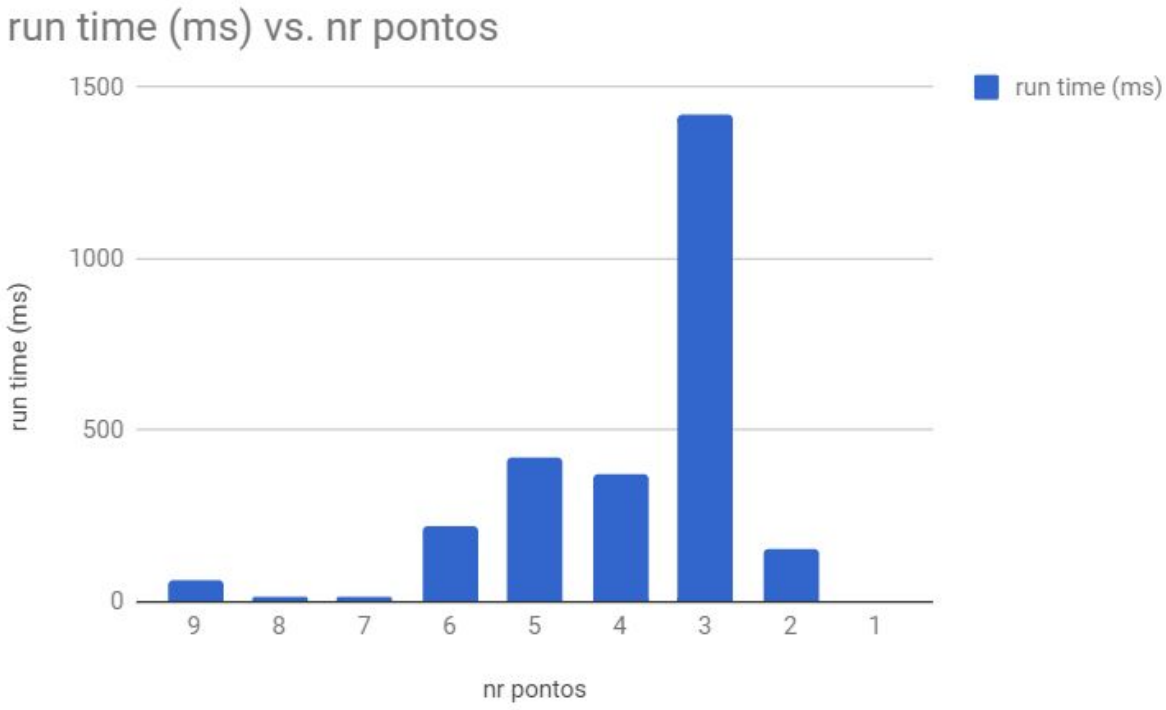
local_inicial(local(0,1,30)).

local_inicial(local(1,2,30)).
local_final(10,1).

capacidade_veiculos(5).

%ponto estratégicos
%local(Coordenada X, Coordenada Y, número de pessoas na carrinha)
ponto_estrategico(ponto(2,1,0)).
ponto_estrategico(ponto(3,2,0)).
ponto_estrategico(ponto(4,3,0)).
ponto_estrategico(ponto(5,4,0)).
ponto_estrategico(ponto(6,1,0)).
ponto_estrategico(ponto(7,2,0)).
ponto_estrategico(ponto(8,3,0)).
ponto_estrategico(ponto(9,4,0)).

Resultados:



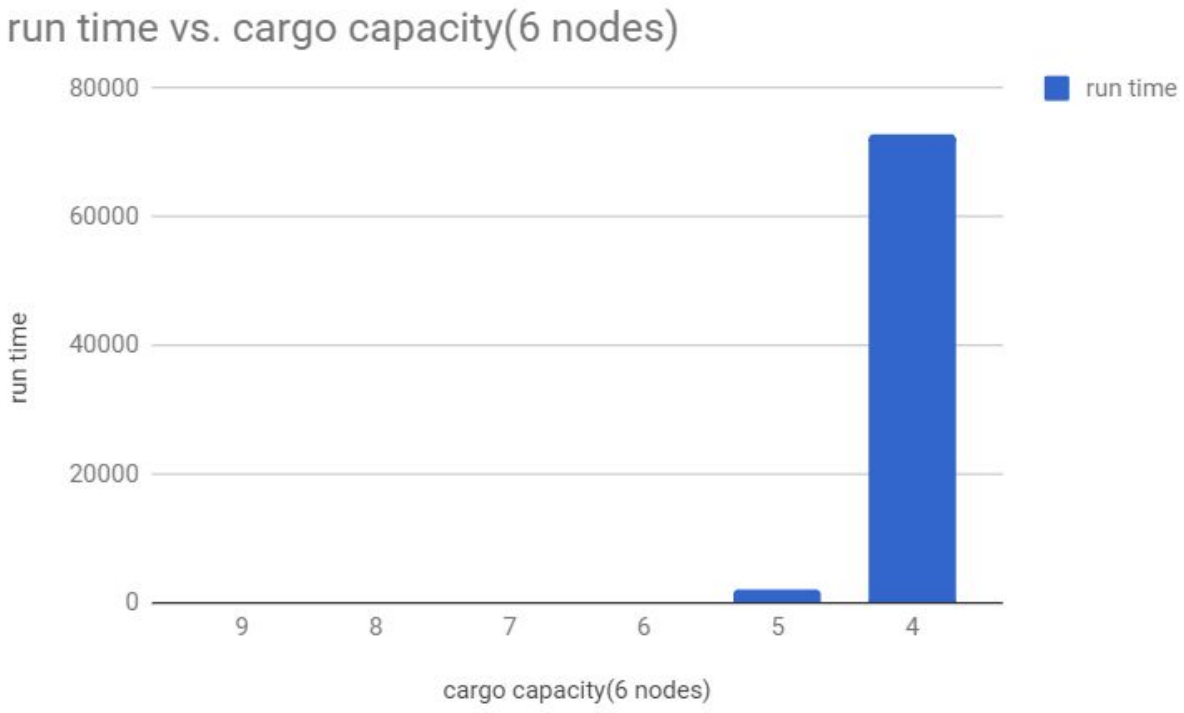
A amostra aleatória diferencia-se na linear na medida da disposição dos pontos, alterando a complexidade temporal do problema. Contudo, a principal conclusão a retirar destes dados é a claro aumento em tempo de execução com um número mais "central" de nós. É de notar também que ambos os exemplo têm um aumento em complexidade aos 3 nós.

2) Alteração das distâncias

A alteração da distância entre os pontos não parece influenciar a complexidade do problema. Contudo, o posicionamento destes tem um grande efeito. No nosso exemplo linear, todos os pontos encontram-se à mesma distâncias dos seus vizinhos. Na amostra aleatória, a distância entre cada ponto e a sua posição tem maior irregularidade, levando ao que o algoritmo tenha de fazer mais decisões.

3) Alteração da capacidade de carga;

De longe o factor mais importante para o tempo de execução do programa.



Como seria de esperar, o tamanho alocado a cada veículo influencia imenso o tempo de execução. Com a redução do tamanho para números inferiores obrigamos cada veículo a fazer viagens extra, o número destas aumentando de forma exponencial com o inverso da capacidade de carga.

Usando exemplo aleatório em 1), as diferenças no tempo de execução do algoritmo são negligíveis até espaço de carga 6. Com a redução deste a partir disso, o programa torna-se muito mais lento, a partir de 3 para abaixo.

Conclusões

Com a intenção de otimizar ao máximo o nosso problema concordamos que a implementação de um algoritmo A* seria o mais indicado. Com esta versão tínhamos como objectivo percorrer uma árvore de decisões da forma mais informada possível, e novamente a escolha de um sistema A* foi adequada.

Com as experiências que realizamos conseguimos também concluir que no nosso problema, o factor mais importante para o tempo de execução é a capacidade de carga de cada vehículo de evacuação. Maximizar o "tamanho" de cada veículo de emergência deverá sempre ser prioridade acima de aumentar o número de pontos estratégicos.

Melhoramentos

Uma possível melhoria seria realizando mais otimizações que permitam melhorar o desempenho de operações pesadas que sejam realizadas com frequência na resolução do problema. Adicionalmente, poderíamos pensar em alguma forma de determinar antecipadamente quando é que um estado já não consegue atingir uma solução válida, permitindo podar a árvore de pesquisa mais cedo.

Outra melhoria seria na função heurística, de modo a que, esta se aproxime ainda mais do valor real, sem comprometer a sua admissibilidade ou a sua eficiência, já que se demorar demasiado tempo a computar, deixa de ser fiável.

Também podiam ser feitas melhorias ao problema com a adição de variáveis como combustível e tempo, está a ser considerado no problema que os veículos utilizados para evacuar turistas contém combustível suficiente para efetuar um número infinito de viagens e que a deslocação dos mesmos do ponto A para o B ocorre instantaneamente.

Finalmente, poderíamos melhorar a interface da aplicação de modo a torná-la mais apelativa e intuitiva.

Recursos

Utilização dos slides teóricos disponíveis na página da disciplina como auxiliar na resolução do problema.

Para a resolução do problema recorremos à implementação do algoritmo A* na linguagem Prolog, utilizando o compilador Sicstus.

Percentagem de trabalho efectivo dos elementos do grupo:

- Ruben Torres: 45%
- Hugo Neves : 27,5%
- Diogo Almeida: 27,5%

Apêndice

Para correr o programa basta compilar o ficheiro solver.pl com o compilador sicstus, após isto basta chamar o predicado *solve_problem.*.

Caso queiramos testar o tempo necessário ao cpu para resolver o problema em questão chamamos o predicado *runtime(X).* em que X é o tempo de execução em milisegundos.

Para resolver outros problemas em ficheiros com nome diferente ao “problem.pl” basta alterar o “:- reconsult(problem).” para “:- reconsult(nome do ficheiro).”, de notar que estes necessitam de seguir as regras indicadas no capítulo Desenvolvimento.