ANÁLISE DE COMPLEXIDADE

Algoritmia Avançada

3º Ano

Instituto Superior de Engenharia do Porto

Grupo 55

João Rodrigues (1210817)
Mateus Fernandes (1210821)
Gabriel Silva (1210808)
Ricardo Freitas (1210828)

Índice de Conteúdos

Índice de Figuras

Introdução

Este estudo da complexidade aborda a otimização de planos de tarefas em ambientes complexos, apresentando um algoritmo inovador que combina técnicas consagradas, como A* e BFS, com Algoritmos Genéticos (AG). Diante da crescente demanda por eficiência operacional em cenários dinâmicos, a proposta visa desenvolver soluções adaptáveis, eficazes e eficientes para a execução de tarefas em edifícios com múltiplos pisos.

O algoritmo proposto integra A* para cálculos de caminhos por pisos, BFS para cálculos entre pisos e AG para a geração de planos de tarefas. A análise de casos de estudo, que variam desde tarefas simples num único piso até desafios complexos envolvendo múltiplos pisos e elevadores. Destaca a eficiência do algoritmo em encontrar soluções ótimas, contribuindo para a compreensão em ambientes dinâmicos.

Algoritmos Genéticos

Para a realização deste estudo foram escolhidos alguns casos que simulam o funcionamento do programa, em casos de alta utilização e em casos de baixa utilização. Com isto, conseguimos perceber, quais são as capacidades dos algoritmos e a sua eficiência, enquanto o programa é utilizado por vários utilizadores ao mesmo tempo.

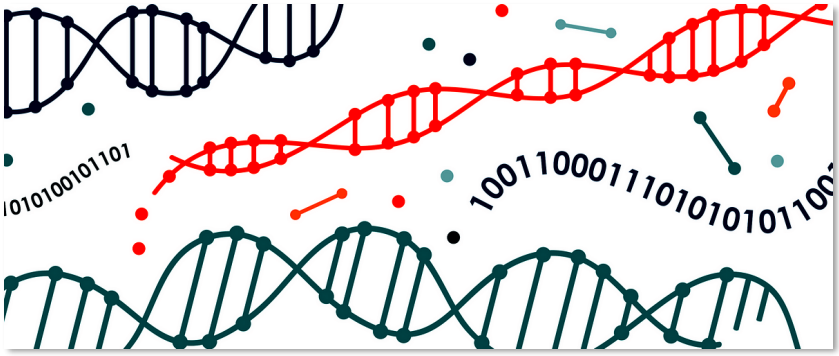

Figura 1- Figura Ilustrativa de Algoritmos Genéticos

Algoritmo desenvolvido

Para ter capacidade de responder a este tipo de requisitos, o algoritmo desenvolvido foi baseado em:

- A*, para cálculo de caminhos por pisos;
- BFS, para cálculo de caminhos entre pisos;
- Algoritmo genético, para geração de um plano de tarefas.

Abordando apenas a vertente de geração de plano de tarefas, o algoritmo genético faz uso dos outros algoritmos anteriores para conseguir perceber o tempo da transição de tarefas e gerar as soluções.

São calculados, estritamente, os caminhos necessários para a geração da sequência. Ou seja, caminhos que se iniciam na origem para todos os outros destinos, excluindo o próprio. Como tal, o tempo de execução deste algoritmo escala de forma considerável dependendo, fortemente, da dificuldade que o A* vai ter de calcular estes "longos caminhos".

Segue na próxima lista os valores parametrizados e utilizados em **todos** os casos de estudo aqui enunciados:

- **Nº gerações máximo**: 100;
- **Nº população**: 5;
- **Tempo de execução máximo**: 5;
- **Nº consecutivo do mesmo melhor indivíduo (estabilização)**: 75;
- **Percentagem de**:
    - **Cruzamento**: 70;
    - **Mutação**: 5;
    - **Pior indivíduo gerar**: 10.

Aleatoriedade no cruzamento entre indivíduos

Para garantir a aleatoriedade no cruzamento dos indivíduos de uma geração, é simplesmente executado um *shuffle* que, como a própria palavra sugere, baralha os indivíduos e compara-os aos pares.

```
% Predicado intermédio que vai garantir que:
%  1 - O melhor indíviduo é passado para a próxima geracao;
%  2 - Nao sao cruzados pares consecutivos (1º e 2º, 3º e 4º...).
% Como?
%  1 - Se se tratar do primeiro índice, ou seja, indivíduo com melhor
prestacao, vai ser diretamente passado para a próxima geracao;
%  2 - É feito um shuffle à lista (excepto o melhor indivíduo).
cruzamento([Ind1*_|Resto], [Ind1|Resto1], 1):-
    !,
    random_permutation(Resto, RestoBaralhado),
    cruzamento(RestoBaralhado, Resto1, 2).

cruzamento([Ind1*_,Ind2*_|Resto],[NInd1,NInd2|Resto1],Index):-
    gerar_pontos_cruzamento(P1,P2),
    prob_cruzamento(Pcruz),random(0.0,1.0,Pc),
    ((Pc =< Pcruz,!,
        cruzar(Ind1,Ind2,P1,P2,NInd1),
      cruzar(Ind2,Ind1,P1,P2,NInd2))
    ;
    (NInd1=Ind1,NInd2=Ind2)),
    Index2 is Index+1,
    cruzamento(Resto,Resto1,Index2).
```

Seleção da nova geração da população do AG

Após fazer uma avaliação sobre a geração N, o algoritmo vai:

- manter **sempre** o **melhor indivíduo** passando-o para a **geração N+1**;
- dar **oportunidade**, ou não, a um indivíduo que em condições normais não teria, de gerar para N+1 (com base na percentagem do pior indivíduo gerar).

Dando esta **oportunidade** é garantido que o algoritmo não é puramente elitista.

```
avalia_populacao([],[]).
avalia_populacao([Ind|Resto],[Ind*V|Resto1]):-
    avalia(Ind,V),
    avalia_populacao(Resto,Resto1).

avalia(Seq,V):-
```

```
    Seq = [PT|T],
    avalia(T,PT,V).

avalia([],_,0).
avalia([Tarefa|Resto],TarefaAnt,V):-
    tarefa2(Tarefa, _, _),
    avalia(Resto,Tarefa,VResto),
```

Parametrização da condição de término

Foram definidas três condições de término:

- Nº de gerações – Existe um número inteiro que define o número máximo de gerações (iterações) que o algoritmo deve gerar;
- Estabilização de solução – Número de vezes consecutivas que o melhor indivíduo se mantém como o melhor avaliado;
- Tempo – Número de segundos que a solução pode atingir.

```
% Numero de geracoes maximo atingido.
gera_geracao(G,G,Pop,_):-!,
    write('Geracao '), write(G), write(':'), nl, write(Pop), nl,
    Pop = [Lista*Tempo|_],
    (retractall(bto(_,_)),!;true),
    asserta(bto(Lista, Tempo)),
    write('Numero de geracoes maximo atingido!').

% Numero de estabilizacao maximo atingido.
gera_geracao(N,_,Pop,[_,_,X]):-
    estabilizacao_solucao(X),
    !,
    %write('Geracao '), write(N), write(':'), nl, write(Pop), nl,
    Pop = [Lista*Tempo|_],
    (retractall(bto(_,_)),!;true),
    asserta(bto(Lista, Tempo)).
    %write('Estabilizacao maxima atingida!').
    % asserta(melhor_individuo()).

gera_geracao(N,_,Pop,_):-
    get_time(Tempo),
    verifica_tempo(Tempo),
    !,
    %write('Geracao '), write(N), write(':'), nl, write(Pop), nl,
    Pop = [Lista*Tempo|_],
    (retractall(bto(_,_)),!;true),
```

```
        asserta(bto(Lista, Tempo)).
        %write('Tempo maximo atingido!').
        % asserta(melhor_individuo()).
```

Adaptação do algoritmo genético para o problema

O algoritmo genético base já solucionava um problema semelhante ao presente, como tal, a adaptação foi ligeira. Existem as seguintes alterações:

- Predicados dinâmicos – O predicado:
    - *tarefa/4* - passa a *tarefa2/3* contendo, por ordem, a designação da tarefa, ponto inicial e ponto de término;
    - *tarefas/1* - mantém-se, contendo o nº de tarefas;
    - *tempo_caminho/3* – contém a tarefa origem, destino e o tempo que leva a deslocar-se;
    - *inicio/1* e *fim/1* – predicados relativos à condição de término de tempo onde por cada geração é verificado se a diferença do que é armazenado em *fim/1* e pelo que é guardado em *inicio/1* é igual ou superior ao tempo máximo permitido;
    - *btoPath/2* - contém o caminho por pisos e o caminho entre pisos, respetivamente (âmbito de LAPR)
    - *ttpl/4* - armazena a tarefa origem, destino, caminho entre pisos e caminho por pisos (âmbito de LAPR);
- Avaliação dos indivíduos –É baseada no tempo que leva a deslocar-se do ponto de término de uma tarefa executada até ao ponto inicial de outra;
- Transmissão da melhor sequência de atendimento de tarefas – Após uma das três condições de término se verificarem, o predicado *bto/2* (sigla para *Best Task Order*), que contém a sequência e o tempo, vai ser *retracted,* se assim for necessário, e *asserted* com os parâmetros relativos ao melhor individuo.
- Obtenção das unidades de tempo entre pontos de acesso – Existem diversos pedidos ao *backend* para definir a base de conhecimentos. Posteriormente, são calculados os caminhos necessários, ou seja, de todos os pontos de término das tarefas para todos os outros pontos de início, excluindo o da própria tarefa. Cálculo

este que é feito através dos predicados feitos no passado *Sprint B* e aproveitando o parâmetro custo que assume que movimentos:

- verticais e horizontais: 1 unidade de tempo;
- diagonais: 1,4 unidades de tempo;
- de transporte:
  - elevador: 30 unidades de tempo;
  - corredor externo: 5 unidades de tempo.

```
tempo_passagens([elev(_,_)|T], Tempo, NTempo):-
  !,
  tempo_passagens(T, Tempo, NTempo2),
  NTempo is NTempo2 + 30. % Tempo por defeito de andar de elevador.

tempo_passagens([cor(_,_,_)|T], Tempo, NTempo):-
  tempo_passagens(T, Tempo, NTempo2),
  NTempo is NTempo2 + 5. % Tempo por defeito de andar no corredor
externo.
```

```
calculo([H|T], Intacta, IndexAct, [H2|T2]):-
  calculo2(H, Intacta, IndexAct, 1, H2),
  Next is IndexAct + 1,
  calculo(T, Intacta, Next, T2).

calculo2(_, [], _, _, []):-!.

calculo2(TarefaAtual, [_|T], Atual, Atual, L):-
  !,
  Next is Atual+1,
  calculo2(TarefaAtual, T, Atual, Next, L).

calculo2(Tarefa, [[TDest, _, Destino]|T], Atual, Ind, [H2|T2]):-

  Tarefa = [TOrig, Origem, _],

  %trace,
  (ponto_acesso(Origem, ColO, LinO, PisoO),!;elev_pos(Origem, ColO, LinO,
PisoO),!;corr_pos(Origem, ColO, LinO, PisoO)),
  (ponto_acesso(Destino, ColD, LinD, PisoD),!;elev_pos(Destino, ColD, LinD,
PisoD),!;corr_pos(Destino, ColD, LinD, PisoD)),

  melhor_caminho_pisos(PisoO, PisoD, Cam, PisosPer),

  node(X1, ColO, LinO, _, PisoO),
  edge(X1, X, _, PisoO),
```

```
  node(Y1, ColD, LinD, _, PisoD),
  edge(Y1, Y, _, PisoD),

  aStar_piso(PisosPer, LC, Cam, X, Y, Tempo),
  eliminate_redundant(LC, LCam),
  %H2 = [TOrig|TDest->Cam*LCam],
  assertz(ttpl(TOrig, TDest, Cam, LCam)),
  tempo_passagens(Cam, Tempo, NTempo),
  asserta(tempo_caminho(TOrig, TDest, NTempo)),
  asserta(tempo_caminho(TDest, TOrig, NTempo)),
  %open('teste.txt', append, Stream),
  %write(Stream, 'CamTempo: '), write(Stream, TOrig), write(Stream, TDest),
write(Stream, NTempo), nl(Stream),
  %close(Stream),
  Ind2 is Ind+1,
  calculo2(Tarefa, T, Atual, Ind2, T2).
```

Caso de Estudo 1

O primeiro caso, trata-se de 3 tarefas a serem executadas todas no mesmo edifício e no mesmo piso, ou seja, um dos melhores casos possíveis, porque não utiliza nem elevadores nem passagens.

Tarefas / Salas	**Origem**	**Destino**
Tarefa 1	D305T	D304T
Tarefa 2	D303T	D302T
Tarefa 3	D304T	D301T

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização do melhor indivíduo e com a seguinte sequência de execução [T1, T2, T3] com uma estimativa de tempo de execução das tarefas de **5.6569** segundos.

Isto significa uma média de **1.8856** segundos necessários para executar cada uma das tarefas pretendidas.

Comentado [RF1]: Estou a adicionar estas palavras porque, como estava, sugeria que era o tempo de execução do algoritmo e não o tempo que o algoritmo estimou que o robô demoraria.

Caso de Estudo 2

O segundo caso, trata-se de 3 tarefas a serem executadas no mesmo edifício, mas com a necessidade de utilizar o elevador para o robot se deslocar entre pisos.

Tarefas / Salas	**Origem**	**Destino**
Tarefa 1	D201T	D301T
Tarefa 2	D305T	D102T
Tarefa 3	D203T	D105T

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização do melhor indivíduo e com a seguinte sequência de execução [T1, T2, T3] com uma estimativa de tempo de execução das tarefas de **50.3134** segundos.

Isto significa uma média de **16,7711** segundos necessários para executar cada uma das tarefas pretendidas.

Caso de Estudo 3

Para este caso, foram consideradas 3 tarefas, mas com necessidade de percorrer vários corredores e elevadores, utilizando os edifícios mais distantes.

Tarefas / Salas	**Origem**	**Destino**
Tarefa 1	APF	D304T
Tarefa 2	D103T	D202T
Tarefa 3	D304T	APF

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização do melhor indivíduo e com a seguinte sequência de execução [T3, T1, T2] com uma estimativa de tempo de execução das tarefas de **44.8995** segundos.

Isto significa uma média de **14.9665** segundos necessários para executar cada uma das tarefas.

Caso de Estudo 4

Para este caso, foram consideradas 3 tarefas, mas com necessidade de percorrer vários corredores e elevadores, utilizando os edifícios mais distantes.

Tarefas / Salas	**Origem**	**Destino**
Tarefa 1	APF	C101T
Tarefa 2	D103T	D202T
Tarefa 3	D304T	D201T

Neste caso, existe a utilização de mais um edifício, comparando com o caso anterior, e existe uma grande diferença de tempo de execução de tarefas. Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização do melhor indivíduo e com a seguinte sequência de execução [T1, T3, T2] com uma estimativa de tempo de execução das tarefas de **99.1421** segundos.

Isto significa uma média de **33.0474** segundos necessários para executar cada uma das tarefas, ultrapassando o necessário para o caso anterior que é bastante semelhante a este.

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização e com a seguinte sequência de execução [T3, T1, T2] com um tempo de execução total de **44.8995** segundos.

Isto significa uma média de **14.9665** segundos necessários para executar cada uma das tarefas pretendidas.

Caso de Estudo 5

Para este caso, foram consideradas 5 tarefas, executadas entre 2 edifícios.

Tarefas / Salas	**Origem**	**Destino**
Tarefa 1	C101T	C105T
Tarefa 2	C103T	C202T
Tarefa 3	C203T	D201T
Tarefa 4	C305T	D304T
Tarefa 5	C203T	C101T

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização e com a seguinte sequência de execução [T4, T3, T2, T5, T1] com um tempo de execução total de **101.9705** segundos.

Isto significa uma média de **20.3941** segundos necessários para executar cada uma das tarefas pretendidas.

Caso de Estudo 6

Para este caso, foram consideradas 6 tarefas, executadas entre 2 edifícios.

Tarefas/Salas	**Origem**	**Destino**
Tarefa 1	C101T	C105T
Tarefa 2	C103T	C202T
Tarefa 3	C203T	D201T
Tarefa 4	C305T	D304T
Tarefa 5	C203T	C101T
Tarefa 6	C310T	C305T

Para este caso, a melhor solução encontrada pelo algoritmo foi através de número de estabilizações máximo alcançado e com a seguinte sequência de execução [T6, T4, T3, T1, T2, T5] com um tempo de execução total de **125.2132** segundos.

Isto significa uma média de **20.8689** segundos necessários para executar cada uma das tarefas. Com isto, conseguimos concluir que o algoritmo apesar de ter alcançado o número máximo de estabilizações definido, conseguiu encontrar uma solução que se enquadra com o caso anterior, especialmente, e com os restantes resultados alcançados sem ter alcançado este mesmo caso.

Caso de Estudo 7

Neste caso foram tratadas 5 tarefas, que se encontravam afastadas.

Tarefas/Salas	**Origem**	**Destino**
Tarefa 1	C101T	D301T
Tarefa 2	C310T	D102T
Tarefa 3	C203T	APF
Tarefa 4	C305T	D304T
Tarefa 5	C203T	C310T

Para este caso, a melhor solução encontrada pelo algoritmo foi através de estabilização e com a seguinte sequência de execução [T3, T1, T2, T4, T5] com um tempo de execução total de **181.9411** segundos.

Isto significa uma média de **36.3882** segundos necessários para executar cada uma das tarefas pretendidas.

Caso de Estudo 8

Neste caso foram requisitadas 15 tarefas, todas elas "perto" umas das outras.

Tarefas/Salas	**Origem**	**Destino**
T1	D305T	D304T
T2	D303T	D302T
T3	D304T	D301T
T4	D202T	D204T
T5	D201T	D201T
T6	D204T	D201T
T7	D205T	D202T
T8	D202T	D201T
T9	D101T	D104T
T10	D102T	D102T
T11	D103T	D105T
T12	D105T	D101T
T13	D105T	D102T
T14	D103T	D101T
T15	D101T	D102T

A solução concebida foi por estabilização da solução com o seguinte plano, "[[T6,T8,T10,T14,T11,T15,T13,T9,T12,T2,T7,T4,T5,T3,T1]*241.8822509939085", e um tempo de execução de **3.89** segundos.

Nota: Foram testados pedidos com mais tarefas, mas devido ao elevado tempo de execução definimos que este será o limite para “tarefas próximas” e os tempos de requisições acima de 15, indefinidos.

Caso de Estudo 9

Neste caso foram requisitadas 12 tarefas, todas elas "afastadas" umas das outras.

Tarefas/Salas	**Origem**	**Destino**
T1	D305T	APF
T2	D303T	APF
T3	D304T	APF
T4	D202T	APF
T5	D201T	APF
T6	D204T	APF
T7	D205T	APF
T8	D202T	APF
T9	D101T	APF
T10	D102T	APF
T11	C204T	APF
T12	C205T	APF

A solução concebida foi por geração máxima atingida e gerou o seguinte plano, [[T11,T7,T6,T4,T8,T9,T10,T3,T2,T1,T5]*962.509667991878", e um tempo de execução de **0.736** segundos.

Nota: Foram testados pedidos com mais tarefas, mas devido ao elevado tempo de execução definimos que este será o limite para "tarefas afastadas" e os tempos de requisições acima de 12, indefinidos.

Conclusão

Os resultados obtidos neste estudo revelam a eficácia do algoritmo proposto na geração rápida de soluções eficientes para uma variedade de casos de estudo. Em grande parte das situações analisadas, o algoritmo demonstrou a capacidade de fornecer respostas rápidas e eficazes, contribuindo para a otimização de planos de tarefas em ambientes complexos.

Ao examinarmos os casos de estudo 8 e 9, identificamos que o cálculo de caminhos desempenha um papel crucial, especialmente em trajetos mais longos. A dificuldade encontrada concentra-se especificamente no cálculo desses caminhos, enquanto o algoritmo genético mantém sua eficiência e eficácia na geração de planos de tarefas.

A complexidade da Figura 3, por exemplo, evidenciou que a estabilização pode ser influenciada pela disposição espacial das tarefas, independentemente da quantidade de pontos envolvidos.

Em resumo, este estudo destaca a eficiência do algoritmo genético na otimização de planos de tarefas, em que os desafios identificados, especialmente em relação aos caminhos mais extensos, apontam para possíveis aprimoramentos futuros.

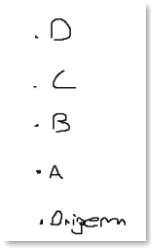

Figura 2 - Pontos em linha.

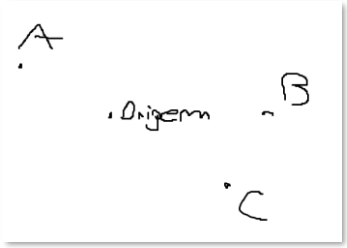

Figura 3 - Pontos dispersos