

Uma imagem com alimentação

Descrição gerada automaticamente

**Relatório de ALGAV**

**3DD \_ Grupo 100**

1191256 André Reis

1191362 Rúben Amorim

**Data: 02/01/2024**

# Geração de todas as sequências de tarefas e escolha da melhor (por permutações)

Gera todas as sequências possíveis com base nas tarefas na base de conhecimento. Cada sequência tem um custo associado (custo da movimentação entre o destino de uma tarefa e a outra), ordenando por ordem crescente de custo. Por fim seleciona a tarefa que esteja no topo da lista.

gera\_permutacoes(MelhorSequencia) :-

findall(Tarefa,tarefa(Tarefa,\_,\_),ListaTarefas),

findall(Seq, permutation(ListaTarefas, Seq), TodasSequencias),

avalia\_populacao(TodasSequencias, TodasSequenciasAvaliadas),

ordena\_populacao(TodasSequenciasAvaliadas, TodasSequenciasAvaliadasOrdenadas),

select(MelhorSequencia, TodasSequenciasAvaliadasOrdenadas, \_).

# Seleção da nova geração da população do AG, garantindo que pelo menos os 2 melhores elementos entre os da geração anterior e os descendentes gerados passem para a geração seguinte, mas que o método não seja puramente elitista.

Para a população gerada este começa por ordenar pelo custo. Seleciona os 20% melhores de N (número de elementos da população) e remove da população geral, passando para a geração seguinte. Nos restantes elementos da população, é associado um produto da avaliação (custo \* nº aleatório entre 0 e 1). Ordena por ordem crescente pelo produto da avaliação e seleciona os N-P primeiros (P é o nº dos melhores elementos selecionados anteriormente). Remove o produto da avaliação aos elementos selecionados no passo anterior. Por fim, junta os elementos selecionados inicialmente com os restantes melhores.

gera\_geracao(N,G,Pop):-

...

seleciona\_melhores(PopOrdenada, Melhores),

remove\_melhores(PopOrdenada, Melhores, PopRestantes),

associa\_produto\_avaliacao(PopRestantes, PopComProduto),

ordena\_populacao\_produto(PopComProduto,PopOrdenadaComProduto),

restantes\_melhores(PopOrdenadaComProduto, Pop, Melhores, RMelhoresComProd),

remover\_produtos(RMelhoresComProd, RMelhores),

append(Melhores, RMelhores, PopNova),

...

seleciona\_melhores(PopOrdenada, MelhoresPop) :-

length(PopOrdenada, T),

P1 is max(1, round(0.2 \* T)),

sublista2(PopOrdenada, 1, P1, MelhoresPop).

restantes\_melhores(PopOrdenadaComProduto, Pop, Melhores, NovaPopulacao) :-

length(Pop, N),

length(Melhores, P),

R is N - P,

sublista2(PopOrdenadaComProduto, 1, R, NovaPopulacao).

# Parametrização da condição de término do AG (pelo menos mais uma para além do nº de gerações)

A cada geração verifica se a geração tem um indivíduo que tem um custo igual ou inferior a um custo ideal dado. Como pode nunca tingir esse custo ideal dado, também verifica se o tempo de execução do algoritmo não ultrapassou o tempo limite dado para a execução.

gera\_geracao(N,G,Pop):-

...

(verifica\_condicao\_termino(Pop, IndivAv), termina(IndivAv), ! ; true),

(verifica\_tempo\_limite(Pop, IndivAv), termina(IndivAv), ! ; true),

verifica\_condicao\_termino(Pop, Individuo\*Avaliacao) :-

solucao\_ideal(SI),

member(Individuo\*Avaliacao, Pop),

Avaliacao =< SI.

verifica\_tempo\_limite(Pop, IndivAv) :-

tempo\_decorrido(TempoDecorrido),

tempo\_limite(Limite),

TempoDecorrido >= Limite,!,

select(IndivAv, Pop, \_).

# Adaptação do Algoritmo Genético para o problema do Planeamento da Trajetória do Robot dentro de edifícios conectados considerando várias tarefas

A adaptação foi que em vez do custo ser o custo da própria tarefa, ser o custo da deslocação de uma tarefa para a outra. Foi feita também a otimização de em vez de o cálculo desse custo ser feito durante a execução do algoritmo, ser calculado o custo de todas as combinações de tarefas antes da execução do algoritmo, sendo gravado numa base de conhecimento dinâmica.

avalia(List, Eval):-

avalia\_aux(List, 0, Eval).

avalia\_aux([\_], Total, Total).

avalia\_aux([T1, T2 | Res], Acc, Eval):-

tarefa(T1, \_, \_),

tarefa(T2, \_, \_),

transita(T1, T2, Eval1),

NewAcc is Acc + Eval1,

avalia\_aux([T2 | Res], NewAcc, Eval).

gera\_transicoes :-

findall((IdOrigem, IdDestino, Avaliacao),

(tarefa(IdOrigem, \_, Destino1),

tarefa(IdDestino, Origem2, \_),

IdOrigem \= IdDestino,

(find\_caminho\_entidades(astar, Destino1, Origem2, \_, \_,

Avaliacao) -> true; fail)),

Transicoes),

retractall(transita(\_, \_, \_)),

assert\_transicoes(Transicoes).

assert\_transicoes([]).

assert\_transicoes([(IdOrigem, IdDestino, Eval) | Resto]) :-

assertz(transita(IdOrigem, IdDestino, Eval)),

assert\_transicoes(Resto).

# Conclusões

Em resumo, este estudo demonstrou a importância da geração de sequências de tarefas por permutações e da seleção criteriosa na evolução de Algoritmos Genéticos para o planeamento da trajetória do robô em ambientes conectados. Ao equilibrar a exploração de novas soluções com a preservação de elementos valiosos, alcançamos avanços significativos. O estudo aprofundado dos métodos atuais e o uso de ferramentas de IA Generativa enriqueceram nossa compreensão, abrindo caminho para melhorias na eficiência do movimento robótico em espaços desafiadores. Este trabalho estabelece bases sólidas para futuras pesquisas, incentivando a busca por estratégias ainda mais eficazes para o planeamento de trajetória do robô em ambientes complexos.