Trabalho Prático nº 2 – Problema de Otimização

Docentes:
Carlos Pereira
Inês Domingues

Trabalho realizado por:
Diogo Oliveira - a2021146037
Lara Bizarro - 2021 130066

Índice

Índice de Imagens

Introdução

No âmbito da cadeira de Introdução à Inteligência Artificial (IIA), foi pedido para realizar um trabalho para resolver um Problema de Otimização, tendo como objetivo conceber, implementar e testar os métodos que encontrem soluções de custo mínimo.

Este dividiu-se em dois componentes, sendo que uma é o componente de criação/implementação e o outro o componente de realização/ análise dos resultados.

No componente da criação/ implementação foram criadas várias funções para posteriormente serem usadas para os testes. O projeto em si, está dividido em três partes, as funções para o Algoritmo da Pesquisa Local, o Algoritmo Evolutivo e o Algoritmo do Híbrido. Ao nível dos testes, foram realizados pelo menos dez testes para cada variância de valores.

Descrição da Função, Objetivo de Otimização, Descrição dos Algoritmos

No primeiro método, abordámos por usar a função de Recristalização, que recebe como parâmetros um ponteiro da solução, um ponteiro da matriz e os vértices, ao usar variáveis definidas gera soluções a cada iteração e vai substituindo a solução anterior no caso de ser melhor.
A função gera_vizinho, é usada para otimizar e encontrar a melhor heurística ao longo dos ficheiros.
A função calcula_fit, como o próprio nome indica serve para calcular o melhor *fitness* de cada ligação.
A função init_dados, é usada para receber os dados dos ficheiros de teste e ir percorrendo estes até chegar ao seu fim.
O objetivo deste primeiro Método é encontrar os menores valores médios com o seu melhor custo associado.

```
custo = calcula_fit( a: sol, mat, vert);
int k = 5;
t = tmax;

while(t > tmin)
{
    for(int i = 0; i<k; i++)
    {
        gera_vizinho( a: sol, b: nova_sol, n: vert);
        custo_viz = calcula_fit( a: nova_sol, mat, vert);

        if(custo_viz <= custo)
            {
            substitui( a: sol, b: nova_sol, n: vert);
            custo = custo_viz;}
        else
            {
            r = rand_01();
                if(r < exp( X: -(custo_viz-custo)/t))
                {
                substitui( a: sol, b: nova_sol, n: vert);
                custo = custo_viz;}
            }
    }
    t *= fa;}
```

Figura 1 - Função de Recristalização

```
void gera_vizinho(int a[], int b[], int n)
{   int i, p1, p2;

    for(i=0; i<n; i++)
        b[i]=a[i];
    do
        p1=random_l_h( min: 0, max: n-1);
    while(b[p1] != 0);
    do
        p2=random_l_h( min: 0, max: n-1);
    while(b[p2] != 1);
    b[p1] = 1;
    b[p2] = 0;}
```

Figura 2 - Função gera_vizinho

```
int calcula_fit(int a[], int **mat, int vert)
{   int i, j, teste, sum = 0;

    for(i = 0; i<vert; i++) {
        teste = 0;
        if (a[i] == 1)
        {
            for (j = 0; j < vert; j++)
            {
                if (a[j] == 1 && mat[i][j] != 0) {
                    sum += mat[i][j];
                    teste = 1;}
            }
                if (teste == 0) {

                    return 999999999;}
        }
    }
    return sum/2;}
```

Figura 3 - Função calcula_fit

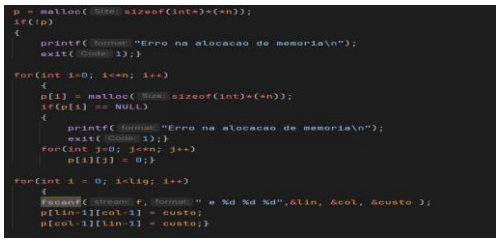

Figura 4 - Função init_dados

No segundo método, tivemos uma abordagem um pouco mais complexa, porque era necessário usar dois tipos de funções, onde uma delas sempre que a ligação não era válida continuava até acabar o ficheiro, enquanto a posterior sempre que a ligação não é válida tenta encontrar outras ligações para a anterior tornar se válida.

A função tournament, serve para fazer a seleção entre pares da população do ficheiro compara o fitness de cada um e o melhor é usado para a próxima geração.

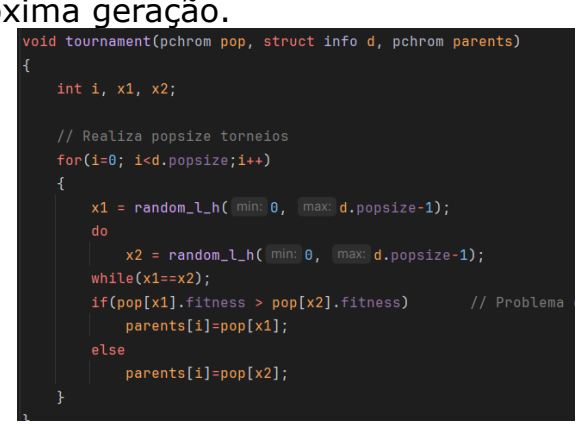

Figura 5 - Função tournament

As seguintes foram criadas para serem mudadas, quando fosse para realizar os testes:

- A função crossover serve para trocar os genes entre os pais, para criar ligações diferentes.

Figura 6 - Função crossover

- A função crossover_uniforme, usando o índice dos genes do país, serve para ligações diferentes, graça a usar o índice torna o programa mais otimizado.

Figura 7 - Função crossover_uniforme

- A função mutation, realiza uma mutação simples ao inverter aleatoriamente os genes de cada filho.

```
void mutation(pchrom offspring, struct info d)
{   int i, j;

    for (i=0; i<d.popsize; i++)
        for (j=0; j<d.numGenes; j++)
            if (rand_01() < d.pm)
                offspring[i].p[j] = !(offspring[i].p[j]);
}
```

Figura 8 - Função mutation

- A função mutation_trade, efetua uma mutação mais específica, ao trocar o valor dos genes por um e por 0 e assim vice-versa, assim obtém se melhorados resultados.

Figura 9 - Função mutation_trade

As seguintes funções são as responsáveis pela avaliação das ligações. A primeira destas a eval_individual_penalizado, ao iterar vai recebendo os genes das ligações, no entanto se estas não encontrarem genes iguais a função segue para outros genes, dado isso os resultados usando esta função não são os mais ótimos.

Figura 10 - Função eval_individual_penalizado

A versão melhorada da função anterior, eval_indivudal_rep, porque ao longo da sua iteração quando encontra genes que não se conectam, começa a procura de genes que correspondam com os anteriores, deste modo, os valores finais são extramente melhores em todos os ficheiros de teste.

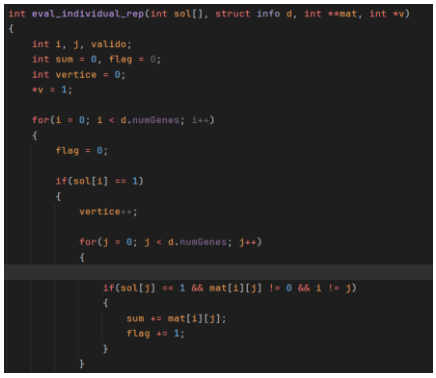

Figura 11 - Função eval_individual_rep

Ao nível do último método, o Híbrido, acaba por combinar os dois últimos métodos supracitada, usando a função gera_vizinho_hibrido, copia a solução atual para a variável "solViz" a solução vizinha e depende do valor definido pelo "TrocaVertice" troca o valor do gene, ou seja inverte o valor de zero para 1 e vice-versa.
A função recristalização_hibrido, uso para cada solução que a função acima devolve, avalia-o usando o eval_individual_rep para calcular o valor de fitness e se o valor do vizinho for menor do atual, substitui por essa solução.

```
void gera_vizinho_hibrido(int sol[],int solViz[],int **mat,int nGenes)
{
    int i;
    //copia a solucao para a vizinha
    for(i=0;i<nGenes;i++) {
        solViz[i] = sol[i];
        if (rand_01() < TrocaVertice) {
            i = random_l_h( min: 0,  max: nGenes - 1);
            solViz[i] = !solViz[i];
        }
        else {
            for (i = 0; i < nGenes; i++) {
                gera_vizinho( a: sol,  b: solViz,  n: nGenes);
            }
        }
    }
}
```

Figura 12 - Função gera_vizinho_hibrido

```
void recristalizacao_hibrido(pchrom pop,struct info d,int **mat){
    int i,j;
    chrom vizinho;
    for(i=0;i<d.popsize;i++)
    {
        for(j=0;j<Ngeneration_h;j++)
        {
            gera_vizinho_hibrido( sol: pop[i].p, solViz: vizinho.p,mat, nGenes: d.numGenes);
            vizinho.fitness=eval_individual_rep( sol: vizinho.p,d,mat, v: &vizinho.valido);
            if(vizinho.fitness < pop[i].fitness)
                pop[i]=vizinho;
        }
    }

}
```

Figura 13 - Função recristalizacao_hibrido

Análise dos Resultados Obtidos

Resultados da Pesquisa Local

Para o primeiro Método, decidimos usar como já referido a Recristalização, uma vez que permite variar mais parâmetros para realizar mais testes.

Durante as experiências, fomos variando o temperatura mínima (tmin), a temperatura máxima (tmax) e a frequência de arrefecimento (fa) e o número de interações.

Verificamos no caso de aceitarmos valores com Custo Igual os melhores valores foram encontrados foi quando a temperatura mínima tinha o seu menor valor e a temperatura máxima tinha o seu valor medio e alto.

1 Vizinhança, Custo Igual - 1000 Iteracoes

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmin	0,1	fa	0,1	52	131,46
Tmax	20	fa	0,5	54	96,06
		fa	0,8	45	70,82
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmin	0,1	tmax	20	54	96,06
fa	0,5	tmax	50	45	68,35
		tmax	70	45	69,44
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmax	50	tmin	0,1	45	68,35
fa	0,5	tmin	0,5	45	76,06
		tmin	1	45	85,02

Figura 14 - Custo Igual

Em relação ao caso de aceitarmos valores com o Custo Menor, encontramos os melhores valores quando a temperatura mínima, a temperatura máxima e a frequência de arrefecimento se encontra no seu valor mais alto.

2 Vizinhança, Custo Menor - 3000 Iteracoes

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmin	0,1	fa	0,1	87	135,58
Tmax	20	fa	0,5	56	109,38
		fa	0,8	52	87,26
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmin	0,1	tmax	20	56	109,38
fa	0,5	tmax	50	56	110,44
		tmax	70	55	111,08
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
tmax	50	tmin	0,1	56	110,44
fa	0,5	tmin	0,5	57	118,58
		tmin	1	61	121,8

Figura 15 - Custo Menor

Ficamos surpresos, quando o usamos a primeira vizinhança obtemos melhores resultados, no entanto pensávamos que iria ser ao contrário.
Algo que não nos surpreendeu foi os valores com o custo igual serem melhores.

Resultados do Algoritmo Híbrido

Para o segundo Método, optamos por fazer três experiências ao longo dos ficheiros. No primeiro, usamos a Avaliação Penalizada, o Tournament, o Crossover e o Mutation.
Durantes estas experiências notamos que os melhores resultados acontecem quando o PopSize se encontra no seu valor máximo e a percentagem de reprodução e a percentagem de mutação estão ambos nos seus valores médios.

Experiencia 1 - EVP_T_C_M

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Prec	0,1	124,27	167
Pmut	0,01	Prec	0,5	118,77	165
		Prec	0,7	118,33	148
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Pmut	0,001	159,2	274
Prec	0,5	Pmut	0,01	118,77	165
		Pmut	0,1	105,27	133
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Prec	0,5	Popsize	100	118,77	165
Pmut	0,01	Popsize	1000	114,07	146
		Popsize	1500	105,17	131

Figura 16 - 1ª Experiência Evolutivo

Na segunda, usamos a Avaliação Penalizada, o Tournament, o Crossover Uniforme e o Mutation.Trade.
Durante estas reparamos que os melhores resultados aparecem quando o PopSize se encontra no valor mínimo, a percentagem de reprodução no valor médio e percentagem de mutação no seu valor máximo.

Experiencia 2 - EVP_T_Cu_Mt

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Prec	0,1	117,03	173
Pmut	0,01	Prec	0,5	117,13	148
		Prec	0,7	121,37	154
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Pmut	0,001	149,93	220
Prec	0,5	Pmut	0,01	117,13	148
		Pmut	0,1	103,13	129
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Prec	0,5	Popsize	100	117,13	148
Pmut	0,01	Popsize	1000	108,6	135
		Popsize	1500	103,63	138

Figura 17 – 2ª Experiência Evolutivo

Na terceira, usamos a Avaliação Reparada, o Tournament, o Crossover e o Mutation.
Durantes estas experiências notamos que os melhores resultados acontecem quando o PopSize se encontra no seu valor máximo e a percentagem de reprodução e a percentagem de mutação estão ambos nos seus valores médios.

Experiencia 3 - EVR_T_C_M					
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Prec	0,1	89	77,73
Pmut	0,01	Prec	0,5	150	128,57
		Prec	0,7	150	127,67
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Pmut	0,001	162	129,27
Prec	0,5	Pmut	0,01	150	128,57
		Pmut	0,1	99	82,9
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Prec	0,5	Popsize	100	150	128,57
Pmut	0,01	Popsize	1000	146	130,9
		Popsize	1500	156	127,5

Figura 18 – 3ª Experiência Evolutivo

Estamos a espera que ao longo dos testes a 3ª Experiência, o contraste em relação à Avaliação Penalizada e Reparada era bem notáveis

Resultados do Algoritmo Híbrido

Para o terceiro Método, como foi ensinado nas aulas práticas, era possível colocar este método em 3 situações, mas havia uma grande diferença a nível de performance para os testes, logo decidimos colocar na primeira para a uma experiência e na primeira e terceira para a última experiência.
Na primeira, usamos a Avaliação Reparada, o Tournament, o Crossover e o Mutation.
Durantes estes testes, reparamos que os melhores resultados acontecem, independente dos valores que íamos mudando.

Experiencia 1 - Hibrido 1º Abordagem

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Prec	0,1	52	50,57
Pmut	0,01	Prec	0,5	55	50,23
		Prec	0,7	55	51
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Pmut	0,001	55	50,53
Prec	0,5	Pmut	0,01	55	50,23
		Pmut	0,1	54	50,73
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Prec	0,5	Popsize	100	55	50,23
Pmut	0,01	Popsize	1000	54	50,3
		Popsize	1500	52	50,1

Figura 19 - 1ª Experiência Híbrido

No entanto, quando colocamos na primeira e na terceira, reparamos que existe uma mudança e os melhores aparecem, quando o PopSize está no seu valor médio e mais alto, a Percentagem de Mutação se encontra no seu valor médio e mais alto e a Percentagem de Reprodução está no seu valor médio.

Experiencia 2 - Hibrido 1ª,3º Abordagem

Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Prec	0,1	153	123,93
Pmut	0,01	Prec	0,5	152	123,17
		Prec	0,7	141	177,7
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Popsize	100	Pmut	0,001	201	152,47
Prec	0,5	Pmut	0,01	152	123,17
		Pmut	0,1	123	104,37
Valores Estáticos		Valores não Estáticos		Melhor Custo	MBF
Prec	0,5	Popsize	100	152	123,17
Pmut	0,01	Popsize	1000	139	114,07
		Popsize	1500	127	109,63

Figura 20 - 2ª Experiência Híbrido

Ficamos surpresos quando a usar só a primeira abordagem do Híbrido obtivemos melhores resultados, em vez de usar a segunda abordagem. No entanto já estávamos a espera que estes testes mostrariam uns valores bons, em relação ao resto, porque o hibrido, refina os valores ao longo da execução

Conclusão

Com base na descrição do trabalho realizado na cadeira de Introdução à Inteligência Artificial (IIA), o estudo visava resolver um Problema de Otimização por meio da conceção, implementação e teste de métodos para encontrar soluções de custo mínimo.
Dividido em dois componentes, o primeiro focou na criação e implementação de funções para posterior teste, enquanto o segundo se concentrou na realização e análise dos resultados.

O componente de criação/implementação consistiu na criação de várias funções destinadas a serem usadas nos testes. O projeto foi estruturado em três partes principais: funções para o Algoritmo da Pesquisa Local, Algoritmo Evolutivo e Algoritmo do Híbrido. Para avaliação, foram realizados pelo menos dez testes para cada variação de valores.

Através dessas etapas, o trabalho proporcionou não apenas a implementação prática dos algoritmos estudados, mas também a oportunidade de testá-los e analisar seu desempenho em relação ao problema de otimização proposto. Os resultados obtidos desses testes forneceram insights valiosos sobre a eficácia e eficiência dos diferentes métodos aplicados, contribuindo para uma compreensão mais aprofundada das abordagens de Inteligência Artificial em cenários de otimização de custos.