Trabalho Prático de Programação Matemática

Alunos:
Nome: Felipe Geraldo de Oliveira. Matrícula: 202310174
Nome: Eduardo Cesar Cauduro Coelho. Matrícula: 202310175

Introdução

Neste trabalho, apresentaremos um problema de otimização relacionado à propagação de fake news e uma proposta de sua solução através da metaheurística Scatter Search. A Scatter Search é uma metaheurística populacional que se baseia em um processo de combinação e melhoria de soluções para resolver problemas complexos de otimização, buscando explorar eficazmente o espaço de soluções, mantendo uma diversidade nas soluções encontradas e, ao mesmo tempo, guiando o processo em direção às melhores soluções possíveis.

Ela se distingue por sua capacidade de integrar soluções promissoras de forma sistemática e gradual, utilizando uma combinação de diferentes estratégias, como recombinação, melhoria local e diversificação, para evitar que o algoritmo se prenda a ótimos locais e conseguir alcançar soluções de alta qualidade.

Apresentação do problema

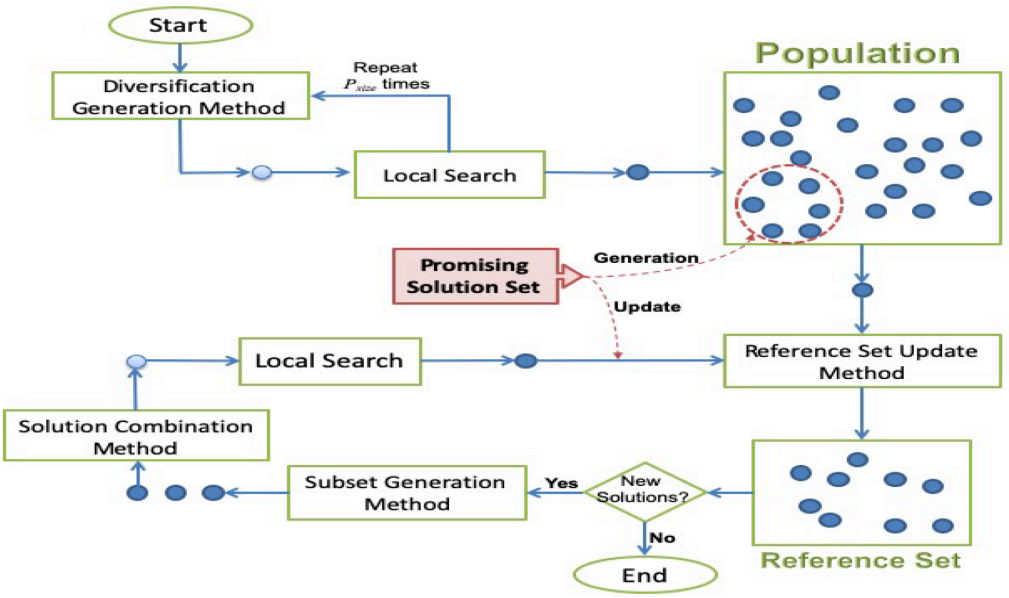
Suponha que uma mensagem classificada como suspeita por meio de uma rede neural seja enviada através de um vértice $s \in V$ de uma rede modelada por um grafo direcionado $G = (V, E)$. O conjunto de vértices V corresponde aos servidores que processam as informações, enquanto os arcos $(u, v) \in E$ são os enlaces de comunicação entre os servidores u, v. Deseja-se reduzir o número de servidores que a notícia falsa pode se encontrar após um período de tempo igual a T.

A propagação de uma informação no enlace $(u, v) \in E$ possui tempo estimado igual a t_{uv}. O Setor de Tecnologia da Informação (STI) do governo federal possui α recursos (softwares auxiliares, por exemplo) tecnológicos que podem ser aplicados aos servidores para retardar o tempo de broadcast da *fake news*. Assim, caso o servidor u possua algum dos α recursos instalado, o tempo de transmissão da *fake news* no enlace (u, v) seria $t_{uv} + \delta$,

em que δ > 0 representa um atraso de propagação. Cada um dos i = 1, ..., α recursos fica disponível no instante β_i.

A solução do problema consiste em determinar a designação dos recursos aos servidores. Cada servidor receberá, no máximo, um recurso. O objetivo definido é a minimização da quantidade de servidores que a *fake news* poderia chegar e um tempo menor ou igual a T.

Framework da Scatter Search



Acima temos uma imagem que representa os módulos utilizados na metaheurística, que serão definidos a seguir:

- Diversification Generation Method: Gera soluções iniciais para a execução do algoritmo. Essas soluções podem ser obtidas através de métodos estocásticos – como uma seleção aleatória dos valores das variáveis – ou através de heurísticas e métodos determinísticos.

- Local Search: Realiza uma busca local visando melhorar as soluções encontradas. Podem ser utilizadas estratégias de first improvement (a primeira melhoria é escolhida) ou best improvement (a melhor melhoria é escolhida).

- Population: Armazena um conjunto de soluções do problema.

- Reference Set: Armazena um conjunto menor de soluções do problema, sendo normalmente um número dez vezes menor que a Population

- Reference Set Update Method: Seleciona as soluções que vão compor o Reference Set. A seleção deve se preocupar com dois fatores: qualidade e diversidade. As soluções devem ser boas e ao mesmo tempo diversas para que haja uma convergência evitando que a solução fique presa em mínimos locais.

- Subset Generation Method: Gera subconjuntos através das soluções contidas no Reference Set. É comum a geração de pares de soluções, dessa forma, são gerados $(n^2-n)/2$ pares para um Reference Set com n soluções.

- Solution Combination Method: Combina os pares de soluções obtidos através do Subset Generation Method, gerando uma ou mais soluções que passarão por um processo de Local Search para que possam ser possivelmente incluídos no Reference Set.

Aplicação da metaheurística na resolução do problema

No problema, as soluções foram definidas como um dicionário onde as chaves são os vértices e os valores são vetores de $|\beta|$ posições, onde $|\beta|$ representa a quantidade de instantes β_i em que um recurso pode ser ativado.

Logicamente, pelas próprias restrições do problema, os vetores ou são completamente formados por 0 ou tem um único número 1 em uma posição i do vetor, indicando que existe um recurso que é ativado no tempo β_i para aquele servidor/vértice

OBS.: nos pseudocódigos a seguir, ‘a <- exemplo’ é uma atribuição simples, enquanto ‘b <= exemplo’ representa a inserção de um item ‘exemplo’ em uma estrutura de dados ‘b’.

A função objetivo do problema é calculada da seguinte forma:

- **Função_Objetivo(solucao):**

```
S <- [ ]
para todo v_inicial ∈ V:
        s <= DijkstraModificado(solucao, v_inicial, interferencias)
C <- [ ]
para todo s ∈ S:
        c <= ContaVertices(s, T)
retorna max(C)
```

Onde ContaVertices conta o número de vértices da solução que são atingidos em tempo menor que T e DijkstraModificado realiza o algoritmo de Dijkstra considerando as interferências que podem atrasar o percurso.

O objetivo do problema é minimizar o resultado obtido por essa função objetivo, ou seja, minimizar o máximo de vértices que podem ser alcançados pela fake news, considerando todos os vértices como possíveis vértices iniciais.

- **Diversification_Generation_Method( ):**

```
samples <- [ ]
solucoes <- [ ]
para todo i ∈ (0, 1, ..., P_size - 1):
        recursos <- rand( )
        enquanto vertices ∈samples:
                vertices <- rand( )
        samples <= vertices
        solucao <= GeraSolucao(recursos)
retorna solucoes
```

Onde GeraSolucao(recursos) gera uma solução conforme o padrão descrito anteriormente através de recursos selecionados randomicamente. Embora omitido no pseudocódigo, cabe ressaltar que rand( ) gera um número específico de vértices respeitando as restrições do problema.

- **Local_Search(solucao, range):**

```
sol_inicial <-  Funcao_Objetivo(solucao)
para todo i ∈ (0, 1, ..., range - 1):
```

```
            v1, v2 <- rand( )
            enquanto v1 = v2:
                  v1, v2 <- rand( )
            solucao.swap(v1, v2)
            nova_solucao <-  Funcao_Objetivo(solucao)
            se nova_solucao < sol_inicial:
                  retorna solucao
            senao:
                  solucao.swap(v1, v2)
      retorna solucao
```

De forma que solucao.swap(v1, v2) significa que os vetores da solução associados ao vértice v1 e vértice v2 são trocados. No final, caso a nova solução não seja melhor, a troca dos vetores é desfeita.

- **Reference_Set_Update_Method(ref_size, prop, Population):**

```
      n_qualid <-  ref_size x prop
      n_divers <- ref_size - n_qualidade
      Reference_Set <- [ ]
      Reference_Set <= Melhores_Solucoes(Population, n_qualid)
      Population <- Population \ Reference_Set
      Reference_Set <= Solucoes_Diversas(Reference_Set,
Population , n_divers)
      retorna Reference_Set
```

Sendo que Population \ Reference_Set indica a diferença entre os dois conjuntos, Melhores_Solucoes retorna as n_qualid melhores soluções da Population considerando o valor da Função_Objetivo e Solucoes_Diversas retorna as n_divers melhores soluções considerando o cálculo do módulo entre a distância dos vetores das soluções restantes e das soluções presentes no Reference_Set. Exemplo:

Distancia entre ({v1: [0, 0, 0, 1], v2: [0, 0, 0, 0]}, {v1: [0, 1, 0, 0], v2: [1, 0, 0, 0]}
=
| [0, 0, 0, 1, 0, 0, 0, 0] - [0, 1, 0, 0, 1, 0, 0, 0] |

Onde é calculada e retornada a norma da subtração dos dois vetores acima.

- **Subset_Generation_Method(Reference_Set):**

```
pares = [ ]
para cada s ∈ Reference_Set:
        para cada r ∈ Reference_Set:
                pares <= (s, r)
retorna pares
```

- **Solution_Combination_Method(pares):**

```
solucoes <- [ ]
para todo par em pares:
        s_inic, s_target <- par
        se Funcao_Objetivo(s_inic) < Funcao_Objetivo(s_target):
                swap(s_inic, s_target)
        dif <- Diferenca(s_inic, s_target)
        solucoes <= Path_Relinking(s_inic, s_target, dif)
return solucoes
```

Onde Diferenca(s_inic, s_target) retorna um vetor que contém os vértices e os respectivos vetores associados ao vértice em relação a cada solução. O objetivo do Path_Relinking é ‘caminhar’ (realizar swaps para diminuir a diferença) do vértice inicial ao vértice target, gerando soluções factíveis intermediárias e retornando a melhor delas segundo o cálculo da Função_Objetivo.

Discussão

Em virtude a utilização de escolhas aleatórias para o povoamento inicial de soluções e o fato da função objetivo ser computacionalmente cara – V x O(E + Vlog(V)), pois o algoritmo de Dijkstra é executado V vezes em cada cálculo da função objetivo, uma considerando cada vértice como inicial – , não conseguimos bons resultados em tempo hábil para a comparação.

Incluímos threads para otimizar o processamento, diminuímos o tamanho da Population mas ainda assim não conseguimos reduzir significativamente o tempo de execução.

Acredito que o uso de heurísticas construtivas iniciais ao contrário de gerar todas as soluções aleatórias foi um elemento que poderia ter diminuído significativamente o tempo de execução e ao mesmo tempo gerar melhores resultados.

Conclusão

Diante do que foi apresentado, considero que o projeto foi bastante útil tanto para o aprendizado teórico do conteúdo de metaheurísticas – em especial a Scatter Search – e otimização de forma geral, tanto para o aprendizado prático, pois os erros e acertos no trabalho foram bastante úteis para possíveis oportunidades posteriores de realizar projetos que envolvem o conteúdo de otimização.

Bibliografia

GLOVER, F.; KOCHENBERGER, G. A. (Ed.). *Handbook of metaheuristics*. 3. ed. Vol. 272. Cham: Springer, 2019.

GLOVER, F.; LAGUNA, M. *Scatter search: methodology and implementations in C*. 1. ed. Vol. 24. Cham: Springer, 2003.