INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUDESTE DE MINAS GERAIS CAMPUS RIO POMBA

CIÊNCIA DA COMPUTAÇÃO

Emiliano Pessata Pereira

João Vitor Ruza Cavalare

Juan Silva Garcia

Leonardo da Mota Melo

IMPLEMENTAÇÃO DE META-HEURÍSTICA PARA O PROBLEMA DO CAMINHO MÍNIMO COM RESTRIÇÃO DE RECURSOS (PCMRC)

Rio Pomba - MG

2025

1. Definição da Solução e Representação

Para resolver o problema do caminho mínimo com restrições, o programa guarda a solução como uma lista de números que representa a ordem dos locais visitados. Para tentar melhorar essa rota, usamos a técnica de Têmpera Simulada com um movimento de corte e reconexão, que tira um pedaço do caminho e tenta ligar o resto de outra forma aleatória. Isso ajuda o computador a não ficar preso em uma única resposta ruim. Para controlar o tempo e a qualidade, definimos uma temperatura inicial de 1000 e um resfriamento de 0.90, além de limitar as tentativas internas a 50 vezes por temperatura. Chegamos a esses números fazendo experimentos práticos, onde notamos que valores maiores faziam o programa demorar muito sem trazer melhorias significativas no custo final, e valores menores davam respostas ruins. O programa para quando a temperatura fica muito baixa (0.1) ou se passar de 2 segundos.

2. Vizinhança e Operadores

A meta-heurística foi implementada para explorar o espaço de busca através de modificações na solução atual. O principal mecanismo utilizado é o operador de corte e reconexão. O algoritmo seleciona aleatoriamente um ponto intermediário no caminho atual, descarta todo o trajeto a partir desse ponto e tenta reconectar ao destino realizando uma nova busca. Essa estratégia permite alterar a estrutura do caminho e evitar que o algoritmo fique preso em soluções repetitivas. Durante esse processo, o sistema calcula imediatamente o consumo de recursos. Caso o novo caminho gerado ultrapasse o limite permitido, aplicamos uma penalidade severa ao valor do custo, o que orienta a busca naturalmente de volta para a região de soluções viáveis que respeitam as restrições.

3. Critério de Parada

Para garantir que o programa não fique rodando para sempre e entregue uma resposta rápida, definimos regras claras. O algoritmo para automaticamente se a temperatura do sistema cair para 0.1 ou se o tempo de processamento passar de 2 segundos. Além disso, ajustamos a temperatura inicial para 1000 e a taxa de resfriamento para 0.90. Escolhemos esses valores depois de fazer vários testes, pois foi a combinação que conseguiu achar as melhores respostas gastando pouco tempo.

4. Tabela de Resultados

Os testes foram realizados utilizando mapas criados automaticamente pelo computador, variando de problemas pequenos a muito grandes. Como esses mapas são únicos, criamos uma comparação interna para medir a eficiência: primeiro, o programa gera um caminho aleatório qualquer (chamado de Inicial), e depois aplica a nossa inteligência para otimizar esse caminho (chamado de Final). A coluna GAP mostra, em porcentagem, o quanto conseguimos melhorar o caminho em relação ao primeiro chute.

Instância	**Nós (N)**	**Inicial**	**Final(SA)**	**GAP(%)**	**Tempo (ms)**	**Status**
inst_pequena	20	66.61	47.00	29,44%	60	Viável
inst_media	50	225.72	127.82	43,37%	55	Viável
inst_grande	100	265.77	127.11	52,17%	254	Viável
inst_muito_grande	200	353.71	166.5	52,91%	108	Viável

5. Análise de Comportamento

Como optamos por utilizar instâncias geradas aleatoriamente para testar a robustez do algoritmo em diferentes cenários, não é possível realizar uma comparação direta com resultados de outros autores, pois os mapas não são os mesmos. Para resolver isso e provar que nosso código funciona, adotamos a estratégia de comparar a nossa Meta-Heurística contra uma solução aleatória simples. Ao analisar a tabela, observamos que o algoritmo foi capaz de reduzir drasticamente o custo das rotas em todos os casos. Nas instâncias maiores, a melhoria chegou a passar de 50%, o que comprova que o método de resfriamento e os operadores de troca estão funcionando corretamente para encontrar caminhos muito mais eficientes do que uma simples busca ao acaso. Além disso, o tempo de execução se manteve baixo, na casa dos milissegundos, demonstrando que a solução é rápida o suficiente para ser usada na prática.