UNIVERSIDADE FEDERAL DE VIÇOSA

DEPARTAMENTO DE INFORMÁTICA

(DPI)

TRABALHO 2 - TRAVELING SALESMAN PROBLEM (TSP) WITH PENALTIES

Pedro Fiorio Baldotto - ES105475

Rafael Zardo Crevelari - ES105468

Disciplina: Metaheurísticas

Professor: Andre Gustavo Dos Santos

31 de Maio 2023

Introdução

Em resumo, o TSPP é uma extensão do TSP tradicional, onde as preferências das cidades em relação à sua posição na rota são consideradas e multas podem ser aplicadas caso essas preferências não sejam atendidas. O objetivo é encontrar a rota que minimize o custo total, levando em conta tanto a distância percorrida quanto às multas impostas pelas cidades.

Neste relatório, será abordado o problema em questão, bem como o conteúdo e os resultados obtidos por meio da aplicação de duas heurísticas baseadas em metaheurísticas diferentes. O problema em foco é a otimização de soluções por meio de heurísticas, utilizando como técnicas vistas em sala de aula.

Serão apresentadas duas heurísticas, cada uma baseada em uma meta-heurística diferente, com o objetivo de encontrar soluções melhores e mais eficientes. Os resultados foram registrados em uma planilha compartilhada pelo professor.

Métodos

Smoothing

O método consiste em simplificar o formato da função objetivo de modo a diminuir o número de mínimos locais. Após fazer repetidas buscas locais, com parâmetros de suavização cada vez menos impactantes, conseguimos gerar uma solução inicial interessante para futuras aplicações.

Pseudocódigo:
Seja **S0** uma solução inicial, **α** a força da suavização, **maiorDistancia** o valor da maior distância entre dois vértices do grafo, **media** o valor médio de distância normalizado:

smoothingMethod(S0, α**):**

S ← S0 // Solução corrente
S* ← S // Melhor Solução

enquanto (α > 1) **faça**
    para (∀e ∈ S) **faça**
        custoS += **distanciaSuavizada**(v1, v2, α) // (v1, v2) = e
    S* = **LS**(S, custoS) // Utilizando a função distanciaSuavizada
    α = α - 1

retornar S*

distanciaSuavizada(v1, v2, α**):**
    d ← **distancia**(v1, v2) // Distância euclidiana
    dn ← d / maiorDistancia

    se (dn >= media)
        retornar media + (dn - media) ^ α
    senão
        retornar media - (dn - media) ^ α

Simulated Annealing

Algoritmo de otimização que busca soluções aproximadas para problemas complexos. Inspirado pelo processo de recozimento de materiais, o algoritmo utiliza uma abordagem probabilística para explorar o espaço de busca. Ele começa com uma solução inicial e faz iterações, gerando novas soluções por meio de perturbações aleatórias. Mesmo soluções piores podem ser aceitas inicialmente, com base em uma função de temperatura que diminui ao longo do tempo. Isso permite escapar de mínimos locais e encontrar soluções ótimas.

Pseudocódigo:
Seja **S0** uma solução inicial, **T0** a temperatura inicial, ***α*** a taxa de resfriamento e **maxIter** o número máximo de iterações:

```
simulatedAnnealing(S0, T0,  α, maxIter):
S ← S0
S* ← S
T ← T0

enquanto (T > 1) faça
        iter ← 0
        enquanto (iter < maxIter) faça
                S' ← vizinho aleatório de S // vizinhança 2-opt
                Δ ← f(S') - f(S)
                se Δ < 0
                        S ← S'
                        se f(S) < f(S*)
                                S* ← S
                senão
                        r ← rand[0, 1]
                        se r < 2.71^(-Δ/T)
                                S ← S'
                iter ← iter + 1
        T ← T * α

retornar S*
```

Busca Tabu

Algoritmo de busca local que utiliza uma lista tabu para evitar retornar a soluções recentemente visitadas. Ele explora diferentes regiões do espaço de busca em problemas de otimização, permitindo escapar de mínimos locais. A lista tabu armazena soluções visitadas e é atualizada ao longo do tempo. A busca tabu é flexível e eficaz para problemas de otimização combinatória.

Pseudocódigo:
Seja **S0** uma solução inicial, **Tmax** o tempo máximo, **tamTabu** o tamanho máximo do tabu.

```
S* ← S
T ← ∅   // Lista Tabu

enquanto (tempo de execução < Tmax) faça
    S' ← melhor N(S) // Fazendo um movimento m (2-opt) tal que m ∉ T, exceto caso m resulte em um S' em que f(S') < f(S*)

    se (size(T) == tamTabu)
        T ← T - T[0] + m
    senão
        T ← T + m

    se f(S') < f(S*)
        S* ← S'

retornar S*
```

Resultados

Em conclusão, os resultados dos testes com os algoritmos SA e TS para o problema do TSPP foram encorajadores. Ambos os algoritmos mostraram-se capazes de encontrar soluções de alta qualidade, com o SA se destacando na capacidade de explorar diferentes regiões da solução e o TS sendo eficiente na busca inteligente por soluções melhores. Essas abordagens heurísticas podem ser aplicadas com sucesso em problemas reais do TSPP, proporcionando resultados satisfatórios.

Busca Tabu:
Utilizamos o método guloso do primeiro trabalho, o método smoothing com a solução trivial e o método smoothing com a solução gulosa para gerar soluções iniciais diversas e aplicar a busca tabu tendo-as como parâmetro. Ao final das buscas, selecionamos a que obteve o melhor resultado. O procedimento é realizado paralelamente.

Em seguida, criamos um conjunto de dados baseado nos parâmetros testados, para chegar a um calibre de parâmetros comum baseado no número de cidades. Inicialmente não usávamos tempo como parâmetro.

	Método	GL+TB+BL	GL+TB+BL	GL+TB+BL	GL+TB+BL	SM+TB+BL	SM+TB+BL	SM+TB+BL
	Parâmetros	1000-100	2000-200	4000-100	5000-75	4 2000-200	5 4000-100	5 2500-75
burma14	zero	30	30	30	30	30	30	30
	cedo	48	48	48	48	48	48	48
	mix	58	58	58	58	58	57	57
berlin52	zero	7542	7542	7542	7542	7542	7542	7542
	cedo	7853	7811	7811	7811	7811	7811	7811
	cedo2	7859	7859	7859	7859	7859	7859	7859
	mix	8014	7949	7905	7937	7905	7934	7949
	mix2	8369	8254	8324	8291	8415	8254	8254
st70	zero	682	677	675	679	680	682	679
	cedo	753	753	746	739	738	742	753
	cedo2	842	847	842	831	846	819	848
	cedo3	810	810	810	809	851	825	825
	mix	840	848	840	838	845	835	835
	mix2	997	913	988	984	968	914	911

gil262	zero		2419	2443	2454		2479	2484
gr666	zero						3254	
dsj1000	zero						19642700	

Após a bateria de testes foi definido:

- Para número de cidades <= 30:
  - Alfa Smoothing: 9
  - Tempo (seg) Busca Tabu: 30
  - Tamanho da Lista Tabu: 50
- Para número de cidades <= 60:
  - Alfa Smoothing: 9
  - Tempo (seg) Busca Tabu: 180
  - Tamanho da Lista Tabu: 100
- Para número de cidades <= 100:
  - Alfa Smoothing: 9
  - Tempo (seg) Busca Tabu: 240
  - Tamanho da Lista Tabu: 125
- Para número de cidades <= 300:
  - Alfa Smoothing: 9
  - Tempo (seg) Busca Tabu: 540
  - Tamanho da Lista Tabu: 150
- Para número de cidades <= 700:
  - Alfa Smoothing: 10
  - Tempo (seg) Busca Tabu: 600
  - Tamanho da Lista Tabu: 225
- Para número de cidades <= 15000:
  - Alfa Smoothing: 10
  - Tempo (seg) Busca Tabu: 3600
  - Tamanho da Lista Tabu: 150

OBS: Para número de cidades <= 15000, aumentar o tempo pode trazer bons resultados, por exemplo, em um teste de aproximadamente 31 horas, conseguimos encontrar 19642700, contudo consideramos esse resultado inviável, pela demora de ser encontrado.

Simulated Annealing:
Utilizamos o método guloso do primeiro trabalho como solução inicial para o simulated annealing. Em nossa implementação, é escolhido um vizinho aleatório da vizinhança 2-opt em cada iteração. Colocamos diversas threads para realizar buscas em paralelo e escolher a que obteve melhor resultado.

Em seguida criamos um conjunto de dados com os parâmetros testados.

Método	SA	SA + BL	SA + BL	SA + BL	SA + BL	SA + BL
Param	300000 - 0.95 - 2000	200000 - 0.95 - 1000	700000 - 0.995 - 500	500000 - 0.99 - 750	500000 - 0.995 - 2000	700000 - 0.995 - 3000
zero	30	30	30	30	30	30
cedo	48	48	48	48	48	48
mix	57	57	57	57	57	57
zero	7596	7542	7764	7809	7542	7711
cedo	7880	7926	7977	8035	7960	8147
cedo2	7911	7859	7935	7999	8015	8090
mix	8014	7974	8025	8308	8020	8096
mix2	8353	8353	8452	8654	8541	8331
zero	696	696	697	688	738	697
cedo	762	762	791	806	806	806
cedo2	869	869	885	885	896	869
cedo3	833	833	888	898	921	947
mix	843	843	881	930	882	891
mix2	935	935	953	1000	1070	1103
zero	2495	2502	2513	2513	2513	2521
zero	3235		3235	3235	3235	3235
zero			20712956	20327483	20424075	20209679

Após a bateria de testes foi definido deixar os parâmetros com os valores:

- Temperatura inicial: 700000
- Taxa de resfriamento: 0,995
- Número de iterações: 3000

Considerações Finais

Em conclusão, os resultados finais do SA e do TS confirmam sua eficácia na resolução do problema do Caixeiro Viajante e do TSPP. Ambos os algoritmos foram capazes de encontrar soluções de alta qualidade e explorar o espaço. Baseado nisso, depois de uma bateria de testes seleta, definimos nossos melhores resultados finais encontrados para os dois métodos, e a porcentagem de um em relação ao outro.

Instância	Multas	TS	SA	Diferença
burma14	zero	30	30	0,00%
	cedo	48	48	0,00%
	mix	57	57	0,00%
berlin52	zero	7542	7711	-2,19%
	cedo	7811	8147	-4,12%
	cedo2	7859	8090	-2,86%
	mix	7905	8096	-2,36%
	mix2	8254	8331	-0,92%
st70	zero	675	697	-3,16%
	cedo	737	806	-8,56%
	cedo2	826	869	-4,95%
	cedo3	805	947	-14,99%
	mix	834	891	-6,40%
	mix2	920	1103	-16,59%
gil262	zero	2445	2521	-3,01%
gr666	zero	3213	3235	-0,68%
dsj1000	zero	19929465	20209679	-1,39%
	Soma Total	19979426	20261258	-1,39%
	Tempo Gasto (min)	135	232	-41,81%

Ao analisar a tabela, podemos concluir que a busca tabu encontra soluções melhores com uma menor quantidade de tempo gasta, em relação ao Simulated Annealing. Além disso, depois de toda a calibração definida, testamos os novos testes com essa calibração, e obtemos os seguintes resultados:

Instância	Multas	TS	SA	Diferença
gil262	cedo	2701	2763	-2,24%
	mix	3277	3356	-2,35%
gr666	cedo	5845	6441	-9,25%
	mix	4420	4494	-1,65%
dsj1000	cedo	20423168	20900280	-2,28%
	mix	20879702	21525197	-3,00%
	Soma Total	41319113	42442531	-2,65%
	Tempo Gasto (min)	191	84	127,38%

Por fim, nesse novo teste, embora o Simulated Annealing tenha rodado em menos tempo, não obteve melhores resultados em comparação a Busca Tabu.

Atividades da Dupla

Durante o projeto, as atividades foram conduzidas em dupla utilizando a técnica de pair-programming. Essa abordagem envolve o acompanhamento simultâneo do desenvolvimento do código, com um membro do grupo observando enquanto o outro codifica, e alternando esses papéis periodicamente. Essa prática garantiu que ambos os membros da equipe adquirissem um conhecimento TOTAL de toda a implementação do projeto.