UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

RELATÓRIO - PROJETO FINAL

HENRIQUE DE ANDRADE FRANÇA	**20210024961**
RODRIGO LANES MENESES	**20210025243**
VITO ELIAS DE QUEIROGA	**20210094483**

JOÃO PESSOA - PB
2023

Apresentação do Trabalho

Este projeto se baseia em um grafo orientado G = (V, A), onde V representa o conjunto de vértices, que consiste no depósito (representado pelo vértice 0) e um número n de clientes. O conjunto A representa os arcos do grafo, que indicam os caminhos que os veículos devem percorrer para atender às demandas dos clientes.

Cada cliente i, identificado no conjunto de vértices, tem uma demanda d_i de pacotes a serem entregues. Para realizar essas entregas, nossa empresa possui uma frota de veículos, todos do mesmo modelo e com uma capacidade máxima Q. É importante observar que todos os veículos devem iniciar e encerrar suas rotas no depósito.

Além das demandas dos clientes e das restrições de capacidade, há outros fatores a serem considerados. Cada arco (i, j) do grafo tem um custo associado c_{ij}, que representa o custo de percorrer esse caminho. Além disso, o uso de cada veículo incorre em um custo r para a empresa. Em situações em que a empresa não pode atender a um cliente com sua frota, existe a opção de terceirizar a entrega, o que implica um custo adicional p_i.

Para otimizar a operação e garantir uma utilização eficiente dos recursos, a empresa estabeleceu uma diretriz importante: pelo menos L entregas devem ser realizadas sem terceirização. Isso significa que a empresa deve atender um número mínimo de clientes com seus próprios veículos.

O objetivo central deste projeto é encontrar um conjunto de rotas que minimize a soma total dos seguintes custos:

1. Custo de roteamento: Isso envolve o custo de percorrer os arcos do grafo para atender às demandas dos clientes.
2. Custo associado à utilização dos veículos: Isso inclui o custo de operação de cada veículo utilizado.
3. Custo de terceirização: Isso se aplica aos casos em que a empresa não pode atender um cliente e deve pagar para que outra empresa realize a entrega.

Em resumo, nosso trabalho visa desenvolver uma solução que otimize a operação logística da empresa, garantindo a satisfação das demandas dos clientes, respeitando as capacidades dos veículos e minimizando os custos globais, incluindo o custo de terceirização. Além disso, nossa solução deve atender à restrição de realizar um número mínimo de entregas sem terceirização, conforme estabelecido pela empresa.

SUMÁRIO

1. ETAPAS

1.1. Implementação do Algoritmo Guloso:

Na etapa de implementação do algoritmo guloso, adotamos uma abordagem que visa iniciar a rota a partir do depósito e, em seguida, selecionar o próximo vértice a ser visitado com base em uma métrica de custo. A métrica utilizada para a escolha do próximo vértice é calculada como a razão entre o custo de ir para um vértice em análise e a demanda do cliente associada a esse vértice. Essa relação ajuda a priorizar a entrega a clientes com maior demanda em relação ao custo associado. O algoritmo continua selecionando o próximo vértice com a menor relação custo/demanda até que a capacidade máxima dos veículos seja atingida. A partir desse ponto, o algoritmo inicia uma nova rota e repete o processo até que todos os clientes sejam atendidos. O resultado desse algoritmo guloso nos fornece uma solução inicial com a qual podemos trabalhar nas etapas seguintes.

1.2. Implementação da 1ª Vizinhança (Movimentos envolvendo uma única rota):

Nesta etapa, implementamos a primeira vizinhança, que envolve movimentos para otimizar rotas individuais. O principal movimento considerado foi o "swap" entre dois vértices em uma rota. A ideia é simular esse movimento, que reduz a complexidade de um “verdadeiro” swap, de O(n^3) para O(n^2) e, ao encontrarmos uma troca que reduz o custo da rota, a mantemos. O "swap" é simulado considerando duas situações:

1.2.1. Vértices Adjacentes: Se os vértices a serem trocados são adjacentes na rota, apenas três arcos são afetados. O custo desses três arcos é recalculado para avaliar a nova rota resultante.

1.2.2. Vértices Não-Adjacentes: Se os vértices não são adjacentes, o custo de quatro arcos é recalculado, incluindo a substituição dos dois arcos entre os vértices pelos arcos que conectam os vértices a seus novos locais.

A seleção de trocas que reduzem o custo é usada para otimizar as rotas individuais. Essa implementação ajuda a melhorar a solução encontrada pelo algoritmo guloso.

1.3. Implementação da 2ª Vizinhança (Movimentos envolvendo múltiplas rotas):

A segunda vizinhança é uma extensão da primeira, mas agora considera a troca de dois vértices entre diferentes rotas “swap inter”. Isso envolve a seleção de dois vértices, um de cada rota, e a simulação de swap entre entre para avaliar se essa troca pode otimizar o custo total das rotas. Como na primeira vizinhança, os cálculos são realizados para verificar se a troca é vantajosa. Essa etapa visa otimizar a alocação de clientes entre veículos, levando a uma redução do custo global.

1.4. Implementação da Terceira Vizinhança (Entregas Terceirizadas):

O objetivo principal aqui é identificar cenários em que a terceirização de entregas pode resultar em uma redução do custo total das rotas. O código itera sobre todas as rotas existentes para explorar essas oportunidades. Dentro do loop externo, percorremos todas as rotas disponíveis. Para cada rota, há um loop interno que percorre vértices dentro da rota.

Inicialmente, é calculado o custo da rota atual. Isso envolve a análise dos custos associados à entrega do cliente em questão, bem como os custos dos arcos que o precedem e o sucedem na rota. O custo é calculado por meio da subtração dos custos dos arcos de chegada e partida e da adição do custo associado à entrega do cliente. Em seguida, é verificado se terceirizar a entrega desse cliente resultaria em uma redução no custo total. Isso é determinado comparando o custo calculado com um melhor custo, que é atualizado sempre que uma opção de terceirização mais vantajosa é encontrada. Se a terceirização resultar em um custo menor, a rota atual é copiada para uma rota auxiliar. Em seguida, o cliente a ser terceirizado é removido da rota auxiliar usando. A lógica por trás disso é que o cliente agora será entregue por outra empresa e, portanto, deve ser removido da rota da empresa original.

Esse processo de avaliação é repetido para todas as rotas e clientes em potencial para terceirização. Ao final da execução, a função retorna a melhor terceirização, que representa a configuração de rotas otimizada, incluindo as entregas terceirizadas que resultam no menor custo total possível.

1.5. Implementação do VND (Variable Neighborhood Descent):

A lógica do código é que, a cada iteração do loop principal, ele verifica se há oportunidades de melhoria dentro das mesmas rotas, entre duas rotas diferentes e por meio da terceirização de entregas. Inicialmente, é criada uma cópia das rotas de entrega e, dentro do loop, o código executa as seguintes etapas:

Primeiramente, ele aplica a 1ª vizinhança a cada rota individualmente, buscando melhorias dentro da mesma rota. Em seguida, verifica se houve uma melhoria nas rotas comparando as novas rotas com as rotas originais. Se uma melhoria foi encontrada, as novas rotas substituem as rotas originais.

Se não houve melhoria nas rotas individuais, o código aplica a 2ª vizinhança, que busca melhorias ao trocar elementos entre duas rotas diferentes. Novamente, é verificado se houve alguma melhoria, e, em caso positivo, as novas rotas substituem as rotas originais.

Finalmente, se as melhorias nas rotas individuais e nas trocas entre duas rotas não resultaram em melhorias, o código aplica a 3ª vizinhança, que procura oportunidades de terceirização de entregas. Como nas etapas anteriores, qualquer melhoria é refletida nas novas rotas.

O processo de busca em vizinhanças é repetido até que nenhuma melhoria adicional seja encontrada. Nesse ponto, o loop principal é encerrado, e as rotas otimizadas, ou seja, as novas rotas, são a saída do algoritmo VND.

2. TABELA DE RESULTADOS COMPUTACIONAIS

	Heurística construtiva				**VND**		
	ótimo	*valor solução*	*tempo*	*gap*	*valor solução*	*tempo*	*gap*
instância **n9k5_A**	428	671	0ns	56.78	609	0ns	42.29
instância **n9k5_B**	506	905	0ns	78.85	843	0ns	66.60
instância **n9k5_C**	559	905	0ns	61.90	843	0ns	50.80
instância **n9k5_D**	408	671	0ns	64.46	609	0ns	49.26
instância **n14k5_A**	471	776	0ns	64.76	685	0ns	45.44
instância **n14k5_B**	565	1058	0ns	87.25	967	0ns	71.15
instância **n14k5_C**	569	1058	0ns	85.94	967	0ns	69.95
instância **n14k5_D**	471	776	0ns	64.76	685	0ns	45.44
instância **n22k3_A**	605	1136	0ns	87.77	811	1ms	34.05
instância **n22k3_B**	777	1652	0ns	112.61	1327	1ms	70.78

instância **n22k3_C**	777	1652	0ns	112.61	1327	1ms	70.78
instância **n22k3_D**	605	1136	0ns	87.77	811	1ms	34.05
instância **n31k5_A**	650	2009	0ns	209.08	1688	1ms	159.69
instância **n31k5_B**	933	3394	0ns	263.78	3073	1ms	229.37
instância **n31k5_C**	939	3394	0ns	261.45	3073	1ms	227.26
instância **n31k5_D**	656	2009	0ns	206.25	1688	1ms	157.32
instância **n43k6_A**	801	2492	0ns	211.11	2205	2ms	175.28
instância **n43k6_B**	1203	4904	0ns	307.65	4617	2ms	283.80
instância **n43k6_C**	1208	4904	0ns	305.96	4617	2ms	282.20
instância **n43k6_D**	802	2492	0ns	210.72	2205	2ms	174.94
instância **n64k9_A**	934	4339	0ns	364.56	4183	3ms	347.86
instância **n64k9_B**	1503	9388	0ns	524.62	9232	3ms	514.24
instância **n64k9_C**	1510	9388	0ns	521.72	9232	3ms	511.40
instância **n64k9_D**	932	4339	0ns	365.55	4183	3ms	348.82
instância **n120k7_A**	1029	5125	0ns	398.06	5002	9ms	386.10
instância **n120k7_B**	2052	12146	0ns	491.91	12023	9ms	485.92
instância **n120k7_C**	2040	12146	0ns	495.40	12023	9ms	489.36
instância **n120k7_D**	1046	5125	0ns	389.96	5002	9ms	378.20

instância **n199k17_A**	1672	16053	1ms	860.11	15665	74ms	836.90
instância **n199k17_B**	3302	43746	1ms	1224.8	43358	73ms	1213.1
instância **n199k17_C**	3301	43746	1ms	1225.2	43358	75ms	1213.5
instância **n199k17_D**	1672	16053	1ms	860.11	15665	77ms	836.90