Capítulo 1

Definição do Problema

1.1 Problema do Caixeiro Viajante (TSP)

Um problema computacional bastante conhecido é o do caixeiro viajante: dado um conjunto de cidades, que devem ser percorridas por um caixeiro, qual a rota que minimiza a distância a percorrer? Um modo direto de resolução deste problema é analisar todas as possibilidades possiveis e encontrar a com menor distância. Esta abordagem não é a mais aconselhavel uma vez que o número de operações requeridas para tal cresce exponencialmente com o número de cidades a percorrer.

Por este motivo se desenvolveu ao longo de varios anos métodos menos custosos para obter ou pelo menos se aproximar da solução. Listaremos agora as estratégias mais comuns utilizadas

1. Vizinho mais próximo

2. Dois ótimo

1.2 Problema de roteamento de veículos

Uma extensão clara do problema do caixeiro viajante é o problema de roteamento de veículos: dado um conjunto de entregas para clientes, que podem ser atendidos por mais que um veículo, qual o melhor conjunto de rotas que miniza a distância total percorrida?

A seguir especificaremos o problema matematicamente.

Dado C (conjunto de clientes) tal que $C \subset \mathbb{R}^2$. O algoritmo $Alg(k, C)$ retorna o conjunto $f_i \in \mathbb{N}_i \times C$ de funções (rotas) tal que $\sum_{i=1}^{k} \#\mathbb{N}_i = \#C$ e $\cap_{i=1}^{k} \mathbb{N}_i = \emptyset$

Dado um conjunto de pontos em um plano devesse determinar a ordem a percorrer estes pontos de modo a minimizar o custo do percurso.

Capítulo 2

Algoritmos de Poupança

2.1 Clark and Wright

O algoritmo econômia de Clarke e Wright é uma das mais conhecidas heuristicas para VRP. Foi desenvolvido em referenciar e aplica-se a problemas para os quais o numero de veículo não é fixo, e trabalha igualmente bem para problemas com digrafos e grafos. Quando duas rotas $(0, \dots, i, 0)$ e $(0, j, \dots, 0)$ podem possivelmente ser mescladas em uma rota simples $(0, \dots, i, j, \dots, 0)$, uma economia de distância $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ é gerada. O algoritmo trabalha como segue:

Step 1. Savings computation

- Calcule o custo $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ para $i, j = 1, \dots, n$ e $i \neq j$.
- Crie n rotas de veículos $(0, i, 0)$ para $i = 1, \dots, n$.
- Ordene as poupanças de modo não crescente (Não é necessário)

Step 2. Melhor combinação possivel

Inicie do topo da lista de custos, executando o seguinte:

- Dada um custo s_{ij}, determine se há duas rotas que podem possivelmente ser mescladas:
  - Uma iniciando com $(0, j)$
  - Uma terminando com $(0, i)$
- Combine essas duas rotas deletando $(0, j)$ e $(0, i)$ e introduzindo (i, j).

2.2 Emparelhamento baseado no algoritmo de poupança

Ista é uma modificação interessante para o algoritmo de poupança padrão (descrições similares são feitas por referênciar e referênciar onde em cada interação a poupança s_{ij} obtida por mesclar rotas p e q é calculado como $s_{ij} = t(S_i) + t(S_j) - t(S_i \cap S_j)$, onde S_k é o conjunto de vértices da rota k, e $t(S_k)$ é o comprimento de uma solução ótima para o TSP em S_k.

Um problema de emparelhamento sobre o conjunto S_k é resolvido usando os s_{ij} valores com custo de emparelhamento, e as rotas correspondem à emparelhamento ótimo estão mescladas a permanencia da fazibilidade.

2.3 Algoritmo de melhoria para Mult-rota

Algoritmos de melhoria esforçam-se para atualizar alguma possivel solução executando um sequência de mudançãs dos vertices e arestas dentro ou entre rotas de veículos. Heuristicas de melhoramento de multi-rotas para VRP operão em cada rota de veículo tomando muitas rotas em um tempo(?). Podemos encontrar descrições de mudanças de arestas para VRP nestas três referências:

- Referenciar
- Referenciar
- Referenciar

Thompson e Psaraftis (1993) propõem um método baseado nos conceitos de k-transferência ciclicas que envolvem transferência de k demandas da rota I^j para a rota $I^{\delta(j)}$ para cada j e inteiro fixo k. O conjunto de rotas $\{I^r\}$, com $r = 1, \ldots, m$, constitui um solução possivel e δ é uma permutação cíclica de um subconjunto de $\{1, \ldots, m\}$. Em particular, quando δ tem cardinalidade fixa C, obtemos um C-ciclo k-transferência. Permitindo k demanda modelo em cada rota, transferência de demanda pode ser realizada através de permutações mais que permutações ciclicas, Devido a complexidade das pesquisas de vizinhanças de transferências cíclicas, é realizada heuristicamente. O operador de 3-ciclo 2-transferências é ilustrado na figura abaixo.

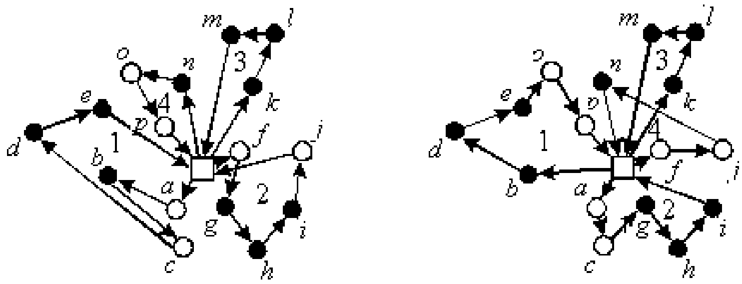
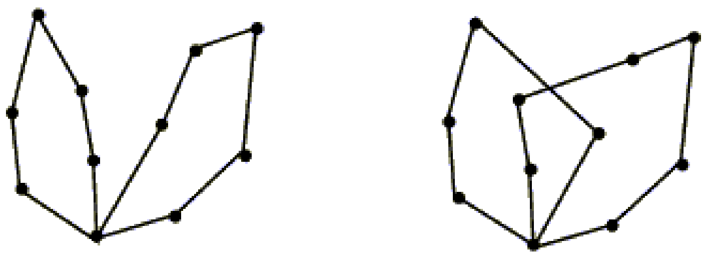


Figura 2.1: O operador de transferência cíclica. A ideia base é transferir simultaneamente os clientes denotados pelo ciclo branco de maneira ciclica entre as rotas. Mais precisamente aqui clientes a e c na rota 1, f e j na rota 2 e o e p na rota 4 são transferidos simultaneamente para as rotas 2, 4, e 1 respectivamente e a rota 3 permanece intocada.

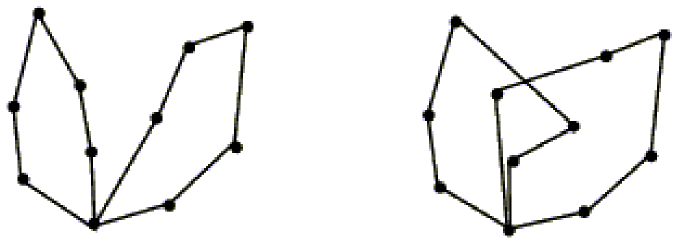
2.3.1 VAN BREEDAM'S ANALYSIS

Agora sumarizaremos a analise de Van Breedam's. Há quatro operações a considerar, as quais são:

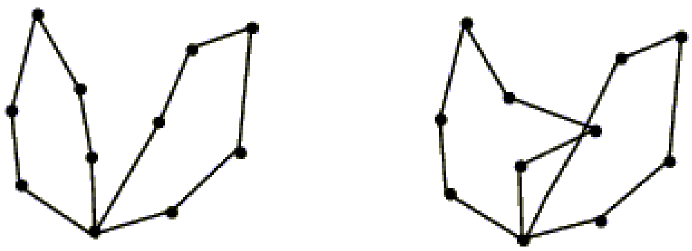
1. String Cross (SC): Duas cadeias de vertices são mudadas pelo cruzamento de duas arestas de diferêntes rotas.
2. String Exchange (SE): Duas cadeias de no mínimo k vértices são mudadas entre duas rotas.
3. String Realocação (SR): Uma cadeia de no mínimo k vertices é deslocada de uma rota para outra, tipicamente com $k = 1$ ou 2.



a) Before b) After



a) Before b) After



a) Before b) After

4. String Mix (SM): O melhor movimento entre SE e SR é selecionado.

   Para avaliar estes movimentos, Van Breedam considerou duas estratégias de melhoria local:

   (a) First Improvement (FI): Consiste de implementar o primeiro movimento que melhora a função objetivo.

   (b) Best Improvement (BI): Avalia todas os possiveis movimentos e implementa o melhor destes.

   Van Breedam então define um conjunto de paramentros que pode influenciar o comportamento da produção da melhoria local:

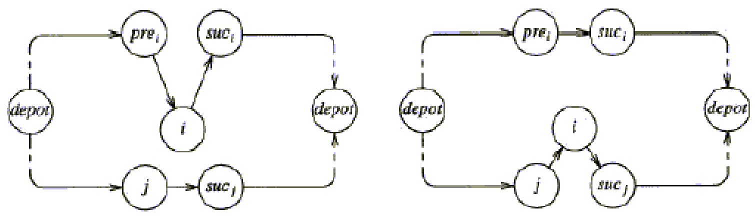
   - A solução inicial (pobre, bom)
   - O comprimenro da string (k) para movimentos do tipo SE, SR, SM (k=1 ou 2)
   - A estratégia de selação (FI, BI)
   - O processo de avaliação para uma string de comprimento $k > 1$ (avaliar todas as possiveis strings de comprimento entre um de rotas, crescentod k

quando uma avaliação de ciclo inteiro tem sido completado sem identificar um mevimento de melhoria).

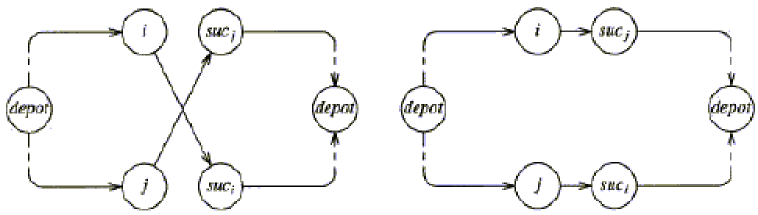
2.3.2 KINDERWATER AND SAVELSBERGH

Na heuristica de Kinderwater e Savelsbergh rotas não são isoladas, então caminhos e clientes são mudados entre diferêntes rotas. A operação que faz essas mudanças são:

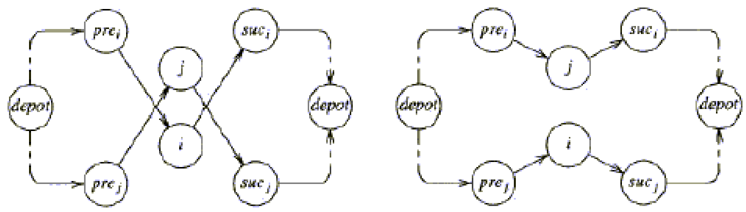
1. Customer Relocation: Um cliente localizado em uma rota é mudado para outra



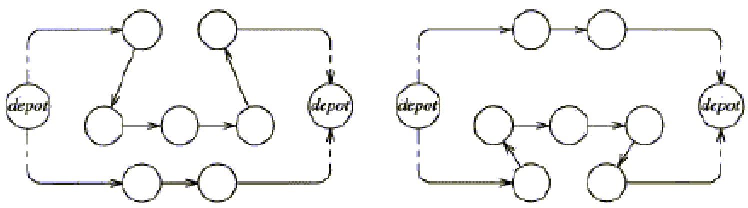
2. Crossover: Duas rotas são misturadas em um ponto



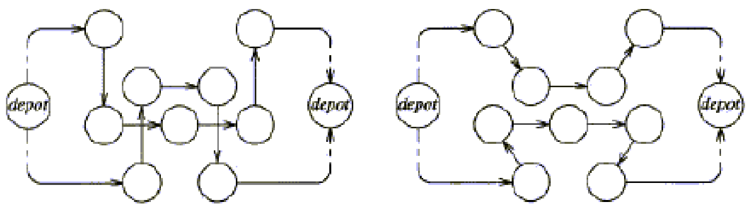
3. Customer Exchamge: Dois clientes de ruas rotas diferêntes são mudados entre duas rotas



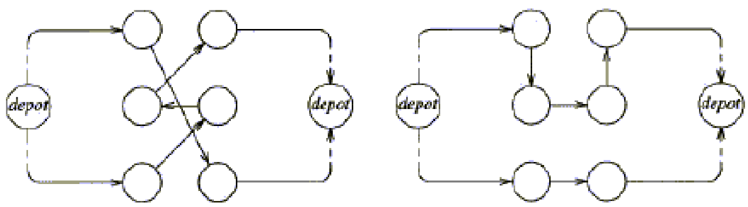
Nas figuras seguintes podemos ver um exemplor um pouco mais complexo:



(a) Relocation of a path.



(b) Exchange of two paths.



(c) A crossover plus 2-exchange.

Capítulo 3

Algoritmos de dois passos

3.1 Cluster-First Route-Second Method

Este método faz uma simples clusterização do conjunto de vértices e então determina uma rota de veículos em cada cluster. Descreveremos o próximo algoritmos:

- Fisher and Kaikumar
- The Petal algorithm
- The Sweep algorithm
- Taillard

3.1.1 Algoritmo de Fisher and Jalkumar

O algoritmo de Fisher e Jaikumar referência é bem conhecido. Resolve um Problema Genérico de Tarefa (GAP) para formar clusters. O número de veículos k é fixo. O algoritmo pode ser descrito como segue:

1. Seed Selection. Escolha pontos semente J_k em V para inicializar cada cluster k.
2. Allocation of Customers to Seeds. Calcule o custo d_{ik} da alocar cada cliente i para cada cliente k com

$$d_{ijk} = \min(c_{0i} + c_{ijk} + c_{J_k 0}, c_{0J_k} + c_{J_k i} + c_{i0}) - c_{0J_k} + c_{J_k 0}.$$

3. Generalized Assignment. Resolver um GAP com custo d_{ij}, peso do cliente q_i e capacidade do veículo Q.
4. TSP Solution. Resolve o TSP para cada cluster correspondente para a solução do GAP.

3.1.2 Algoritmo Pétala

Uma extensão natural do algoritmo de varredura é gerar muitas rotas, chamadas pétalas referência, e fazer uma seleção final resolvendo um problema de partição de conjunto da forma:

$$\min \sum_{k \in S} d_k x_k$$

sujeito a:

$$\sum_{k \in S} a_{ik} x_k = 1 \quad (i = 1, \dots, n)$$

com

$$x_k = 0, 1, k \in S,$$

onde S é o conjunto de rotas, $x_k = 1$ se, e somente se, a rota k pertence à solução, a_{ik} é o parâmetro binário iqual a 1 somente se o vértice i pertence a rota k, e d_k é o custo da rota da pétala k. Se as rotas corresponde a setores contínuos de vértices, então esse problema possui o propriedade de coluna circular e é resolvido em tempo polinomial referência.

3.1.3 O Algoritmo de varredura

O Algoritmo de varredura aplica-se a instâncias planares do VRP. Consiste de duas partes:

- Split: Clusters possiveis são iniciados formando rotação e centro de raio no deposito
- TSP: Uma rota de veículo é então obtida para cada cluster resolvendo o TSP.

Algumas implementações incluem uma fase de otimização posterior na qual os vértices são mudados entre clusters adjacêntes, e as rotas são reotimizadas. Uma implementação simples deste método é como segue. onde assumimos que cada vértice i é representado por suas coordenadas polares (θ_i, ρ_i), onde θ_i é o ângulo e ρ_i é o comprimento de raio. Atribuindo um valor $\theta^* = 0$ para um vértice arbitrário i^* e calculando os ângulos remanecentes de IMG6$(0, i^*)$. Graduando os vértices em ordem crescente de sua IMG2.

1. Route Initialization. Escolha um não utilizado veículo k.
2. Route Construction. Inicie dos vértices não roteados tendo ângulo menor, atribuindo vertices ao vértice k enquanto sua capacidade ou o comprimento máximo da rota não é excedido.
3. Route Optimization. Otimizar cada rota de veículos separadamente resolvendo o correspondete TSP (exatamente ou aproximadamente).

3.1.4 Algoritmo de Talliard

O algoritmo de Talliard referência define-se vizinhança usando o λ-intermudança mecanismo de Geração referência. Rotas individuais são reotimizadas usando algoritmo de otimização de referência. Um traço nobel do algoritmo de Talliard é a decomposição dos problemas essenciais em subproblemas.

Em problemas planares, estes subproblemas são obtidos inicialmente particionando os vértices em setores centradas no deposito, e em regiões concentricas dentro de cada setor. Cada subproblema pode ser resolvido independentemente, mas movimentos periódicos de vértices para setores adjacêntes são necessários. Isso faz sentido quando o deposito é centrado e os vértices são uniformemente distribuidos no plano.

Para problemas não planares, e para problemas planares não possuindo esta propriedade, o autor sugere uma método de partição diferênte baseado no calculo da mais curta espação de arborecência enraizadas no depósito. Esse método de decomposição é particularmente bem conviniente para implementeções paralelas com subproblemas podem então ser distribuidos através de varios processadores.

A combinação dessas estratégias produz excelentes resultados computacionais.

3.2 Métode de roteamento seguido de cluster

O métode de roteamento seguido de cluster constroi numa primeira fase uma grande rota de TSP, observando as restrições laterais, e depositando esta rota dentro das possiveis rotas de veículos numa segunda fase. Essa ideia aplicasse a problemas com uma número livre de veículso. Foi primeiro posto por Beasley o qual observou que o problema da segunda fase é um problema de caminho mínimo padrão em grafo acíclico e pode ser assim resolvido em tempo $O(n^2)$. No algoritmo de caminho mínimo, o custo d_{ij} da viagem entro os nós i e j é igual a $c_{0i} + c_{0j} + l_{ij}$, onde l_{ij} é o custo da viagem de i a j na rota TSP.

Não estamos conscientes de algumas experiências computacionais mostrando que a heuristica de roteamento primeiro seguido de cluster são competitivas com outras aproximações.