4
Problema de Roteamento de Veículos com Janela de Tempo

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) consiste em uma frota de veículos idênticos, denotado por K, um conjunto de clientes V e um grafo direcionado G. O grafo contém $|V| + 1$ vértices, onde os clientes estão localizados nos vértices $1, 2, ..., n$ e o depósito é representado pelo vértice 0. O conjunto de arcos A representa os caminhos diretos entre o depósito e os clientes e entre os clientes. Cada arco (i, j), onde $i \neq j$, tem um custo c_{ij} e um tempo de viagem t_{ij} associados.

Cada veículo tem capacidade C e cada cliente i tem uma demanda d_i associada. Cada cliente tem também uma janela de tempo $[a_i, b_i]$, que representa o intervalo de tempo em que o veículo deve realizar o serviço no cliente i e um tempo de serviço s_i. O tempo de viagem t_{ij} pode conter o tempo de serviço no cliente i. O veículo tem que chegar em i antes de b_i. É permitido chegar no cliente antes de a_i, porém o veículo deverá ficar parado esperando a abertura da janela de tempo. O depósito tem uma janela de tempo $[a_0, b_0]$. Os veículos não podem sair do depósito antes de a_0 e devem retornar ao depósito antes de b_0.

O objetivo do problema é determinar o conjunto de rotas de menor custo, uma para cada veículo, tal que:

- Cada rota comece e termine no depósito
- Cada cliente em V seja visitado apenas uma vez
- A demanda total dos clientes visitados em uma rota não pode exceder a capacidade do veículo
- O serviço em cada cliente deve ser realizado dentro de suas janelas de tempo

4.1 Trabalhos Existentes

Métodos exatos para o PRVJT usam uma das duas técnicas: geração de colunas ou relaxação lagrangeana. Desrochers et al (1992)[4] utilizaram técnicas de geração de colunas para resolver na otimalidade muitos problemas de 50 clientes e alguns de 100 clientes, minimizando a distância total percorrida. A idéia básica é formular o PRVJT como um problema de partição de conjuntos considerando todas as rotas viáveis implicitamente. Como este número cresce exponencialmente com o número de clientes, apenas uma pequenas fração dessas rotas são incluídas no modelo. A cada iteração, o algoritmo checa se existe uma rota ainda não incluída no modelo que reduz a distância total percorrida. Isto pode ser feito resolvendo o problema de caminho mais curto com janela de tempo e restrição de capacidade, PCMCJTRC. Se tal rota existir, a coluna correspondente é adicionada ao modelo. Este procedimento é repetido até que a solução ótima para o conjunto de rotas seja encontrada. Kohl[5] e Kohl e Madsen (1997) utilizaram a relaxação lagrangeana para remover a restrição que garante que todos os clientes sejam visitados. Isto permite decompor o problema em subproblemas de caminho mais curto com janela de tempo e restrições de capacidade para cada veículo. Apesar de ser possível encontrar boas soluções heurísticas para estes subproblemas relativamente fáceis, encontrar os multiplicadores Lagrangeanos ótimos requer um esforço computacional muito grande. Problemas de até 100 clientes puderam ser resolvidos por este método.

Padberg e Rinaldi (1991) foram os primeiros a desenvolverem métodos com planos de cortes para resolver instâncias do PCV. Araque et al. (1994) desenvolveram um algoritmo de *branch-and-price* para o problema de roteamento de veículos com clientes idênticos. Augerat (1995) apresentou algumas desigualdades válidas novas para o problema de roteamento de veículo com restrições de capacidade simétrico. No *branch-and-cut*, o problema é inicialmente formulado como um *MIP*. As restrições de integralidade são relaxadas e o problema linear resultante é resolvido. Se a solução é inteira, o algoritmo termina e a solução é ótima para o problema original. Se a solução é fracionária, desigualdade válidas, ou cortes, são adicionados ao modelo e o procedimento é repetido. Quando as rotinas de separação não puderem identificar mais nenhum corte, ou quando a adição de uma nova restrição não causa nenhuma mudança no problema, é feito o *branch*.

4.2 Formulação do Problema

Na formulação mais clássica do PRVJT, o problema pode ser modelado como um problema de redes multi-fluxos com restrições de janela de tempo e capacidade. Esta formulação envolve dois tipos de variáveis: as variáveis de tempo $s_{ik}, i \in V, k \in K$ que especificam o instante de tempo em que o veículo k iniciou o serviço no cliente i e as variáveis de fluxo $x_{ijk}, (i,j) \in A, k \in K$, tal que:

$$x_{ijk} = \begin{cases} 1 & \text{, se o veículo } k \text{ vai do cliente } i \text{ para o cliente } j \\ 0 & \text{, caso contrário} \end{cases}$$

$$\text{(PRVJT)} \begin{cases} Z_{PRVJT} = \min \sum_{k\in K} \sum_{i\in V} \sum_{j\in V} c_{ij} x_{ijk} & (1) \\ \text{sujeito a} & \\ \sum_{k\in K} \sum_{j\in V} x_{ijk} = 1 & ,\forall i \in V \quad (2) \\ \sum_{i\in V} d_i \sum_{j\in V} x_{ijk} \leq C & ,\forall k \in K \quad (3) \\ \sum_{j\in V} x_{0jk} = 1 & ,\forall k \in K \quad (4) \\ \sum_{i\in V} x_{ihk} - \sum_{j\in V} x_{hjk} = 0 & ,\forall h \in V\backslash\{0\}, \forall k \in K \quad (5) \\ \sum_{i\in V} x_{i0k} = 1 & ,\forall k \in V \quad (6) \\ s_{ik} + t_{ij} - M(1 - x_{ijk}) \leq s_{jk} & ,\forall i,j \in V, \forall k \in K \quad (7) \\ a_i \leq s_{ik} \leq b_i & ,\forall i \in V, \forall k \in K \quad (8) \\ x_{ijk} \in \{0,1\} & ,\forall i,j \in V, \forall k \in K \quad (9) \end{cases}$$

A função objetivo (1) expressa o custo total. A restrição (2) garante que cada cliente seja visitado apenas uma vez, (3) garante que a capacidade do veículo nunca é excedida. As equações (4), (5) e (6) caracterizam o fluxo em um caminho a ser seguido por um veículo k, e garantem que cada veículo deixa o depósito, que ao chegar em um cliente, o veículo também sai deste cliente e finalmente retorna ao depósito, respectivamente. A inequação (7) garante que o veículo k não pode chegar em j antes de $s_{ik} + t_{ij}$ se está viajando de i para j. A restrição (8) garante que a janela de tempo está sendo observada e (9) são as restrições de integralidade.

4.2.1 Formulação Proposta

A formulação mestre aqui proposta para o PRVJT coincide com a do PRVC definida na seção 3.1.3, de modo que possamos utilizar o mesmo algoritmo *branch-and-cut-and-price* apresentado no capítulo anterior.

Assim, de forma a contemplarmos as restrições de janela de tempo, que não estão presentes em PRVC temos que alterar apenas o subproblema de geração de colunas do algoritmo, de modo que as colunas (rotas) geradas respeitem não só a capacidade dos veículos, mas também as janelas de tempo dos clientes.

Como o subproblema de geração de colunas é o responsável pela geração das q-rotas sem 2-ciclos, alterando a definição das q-rotas para uma rota $\{0, v_1, ..., v_r, 0\}$ tal que $\sum_{i=1}^{r} d(v_i) \leq C$ e $a_i \leq s_i \leq b_i$, onde s_i é o instante de tempo em que o serviço é realizado no cliente v_i, $v_i \neq 0$ para cada i em $\{1, ..., r\}$ e $r \geq 1$, conseguimos adaptar a formulação do PRVC para o PRVCJT.

Encontrar as q-rotas que respeitam as restrições de capacidade e de janela de tempo também pode ser feito em tempo pseudo-polinomial, $O(n^2CT)$, onde T é a largura da janela de tempo. Assim como no PRVC a eliminação de 2-ciclos pode ser feita sem alterar a complexidade do subproblema.

4.2.2 Pré-processamento

Redução das Janelas de tempo

A complexidade dos algoritmos propostos para a solução dos PRVJT é em função da largura das janelas de tempo. Para melhorar a eficiência desses algoritmos, a largura das janelas de tempo podem ser reduzidas a priori aplicando uma série de regras descritas em Desrochers, Desrosiers e Solomon[4].

Em cada nó a redução das janelas de tempo é feita aplicando as condições teste mostrados abaixo:

- Ajuste da abertura da janela de tempo pelo instante de tempo mais cedo que é possível chegar no nó k vindo de seus predecessores

  Para fazermos tal ajuste, determinamos o instante de tempo mais cedo que é possível chegar no nó k vindo de um predecessor i, ou seja, se

sairmos de i na abertura de sua janela de tempo, chegaremos em k no instante $a_i + t_{ik}$, conforme mostrado na figura 4.1. Repetindo esse procedimento para todos os predecessores de k, o instante de tempo mais cedo que é possível chegar em k vindo de seus predecessores é dado por:

$$\min_{(i,k)\in A}\{a_i + t_{ik}\},$$

e a abertura da janela de tempo do cliente k pode ser ajustada para:

$$a_k = \max\{a_k, \min_{(i,k)\in A}\{a_i + t_{ik}\}\}.$$

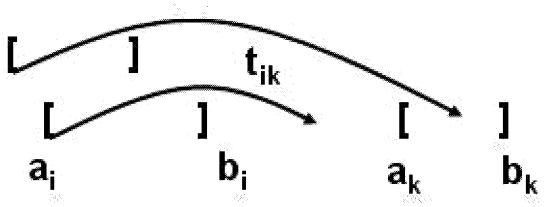

Figura 4.1: Ajuste da abertura da janela de tempo em função da janela de tempo de seus predecessores

- Ajuste do fechamento da janela de tempo pelo instante de tempo mais tarde que é possível chegar no nó k vindo de seus predecessores.

  Para fazermos tal ajuste, determinamos o instante de tempo mais tarde que podemos chegar no nó k vindo de um predecessor i, ou seja, se sairmos de i no fechamento de sua janela de tempo, chegaremos em k no instante $b_i + t_{ik}$, conforme mostrado na figura 4.2. Repetindo esse procedimento para todos os predecessores de k, o instante de tempo mais tarde que é possível chegar em k vindo de seus predecessores é dado por:

$$\max_{(i,k)\in A}\{b_i + t_{ik}\},$$

  e o fechamento da janela de tempo do cliente k pode ser ajustada para:

$$b_k = \min\{b_k, \max_{(i,k)\in A}\{b_i + t_{ik}\}\}$$

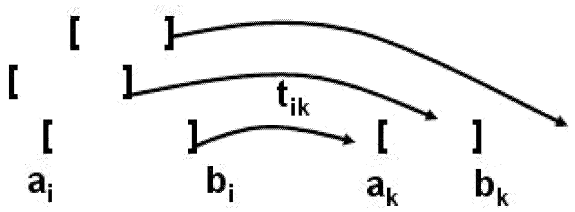

Figura 4.2: Ajuste do fechamento da janela de tempo em função da janela de tempo de seus predecessores

- Ajuste da abertura da janela de tempo pelo instante de tempo mais cedo que é preciso sair do nó k para os seus sucessores.

  Sabendo que o instante de tempo mais cedo que podemos iniciar o serviço no nó j, sucessor do nó k, é na abertura de sua janela de tempo, podemos então sair do nó k no instante de tempo $a_j - t_{kj}$ de forma a chegarmos em j no exato instante que a janela de tempo está abrindo, conforme a figura 4.3. Se repetirmos esse processo para todos os nós sucessores ao nó k, determinamos que o instante de tempo mais cedo que é preciso sair de k para chegar em seus sucessores antes da abertura de suas janelas de tempo é dado por:

$$\min_{(k,j)\in A}\{a_j - t_{kj}\},$$

  e a abertura da janela de tempo do cliente k pode ser ajustada para:

$$a_k = \max\{a_k, \min_{(k,j)\in A}\{a_j - t_{kj}\}\}$$

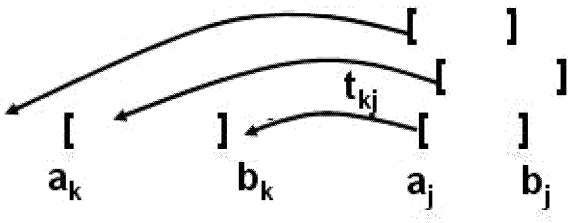

Figura 4.3: Ajuste da abertura da janela de tempo em função da janela de tempo de seus sucessores

- Ajuste do fechamento da janela de tempo pelo instante de tempo mais tarde que é preciso sair do nó k para os seus sucessores

  Sabendo que o instante de tempo mais tarde que possível no nó j, sucessor do nó k, é no fechamento de sua janela de tempo, é preciso

sair do nó k no instante de tempo $b_j - t_{kj}$ de forma a chegarmos em j no exato instante que a janela de tempo está fechando, conforme a figura 4.4. Se repetirmos esse processo para todos os nós sucessores ao nó k, determinamos que o instante de tempo mais tarde que é preciso sair de k para chegarmos em todos os seus sucessores antes do fechamento de suas janelas de tempo é dado por:

$$\max_{(k,j)\in A}\{b_j - t_{kj}\},$$

e o fechamento da janela de tempo do cliente k pode ser ajustada para:

$$b_k = \min\{b_k, \max_{(k,j)\in A}\{b_j - t_{kj}\}\}$$

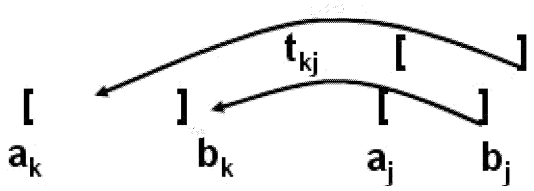

Figura 4.4: Ajuste do fechamento da janela de tempo em função da janela de tempo de seus sucessores

Os arcos que se tornarem inviáveis são removidos do conjunto de arcos A. Após examinar todos os nós, deve-se iniciar uma nova iteração. Na prática após duas ou três iterações não existem mais reduções a serem feitas.

4.3 Algoritmo *branch-and-cut-and-price*

O algoritmo *branch-and-cut-and-price* robusto para o PRVC apresentado por Fukasawa et al. [37], combina um algoritmo *branch-and-cut* com um algoritmo de geração de colunas que correspondem às q-rotas definidas no capítulo 3. A formulação mestre explícita (*ME-PRVC*) para o PRVC, com número exponencial de restrições e variáveis, é descrita na seção 3.1.3.

A especialização do algoritmo para o PRVJT consiste na modificação do esquema de geração de colunas para forçar as q-rotas a visitar os clientes dentro de suas janelas de tempo.

Nas seções 4.3.1 a 4.3.3 são descritas os principais componentes desse algoritmo: a geração de cortes, a geração de colunas e o algoritmo *branch-and-bound*.

4.3.1 Geração de cortes

Tanto o algoritmo original quanto essa especialização consideram inicialmente na formulação utilizada (*DWM-PRVC*) somente restrições relativas aos graus dos vértices e cortes violados são adicionados à formulação quando necessário.

Os cortes de capacidade não são os únicos que podem aparecer nessas formulações. Um corte genérico $\sum_{e \in E} a_e.x_e \geq b$ pode ser incluído como $\sum_{j=1}^{p}(\sum_{e \in E} a_e.q_j^e).\lambda_j \geq b$. Assim, o algoritmo adiciona todas as classes de cortes descritos em [57] (*framed capacity*, *strengthened comb*, *multistar*, *partial multistar*, *generalized multistar* e *hypotour cuts*), utilizando o pacote de rotinas *CVRPSEP* [56] para separar tais cortes. Esses cortes são definidos em termos das variáveis x_e^*.

4.3.2 Geração de Colunas

O subproblema exato de geração de colunas para a formulação por número exponencial de variáveis do PRVJT é equivalente ao problema de caminho mais curto elementar com janela de tempo e restrição de capacidade, ou PCMCEJTRC, isto é, o caminho mais curto que sai do depósito e retorna ao depósito, visitando os clientes da rota apenas uma vez e que em momento algum viola as janelas de tempo e a restrição de capacidade.

Infelizmente, não é conhecido nenhum método eficiente para resolver o PCMCEJTRC. De fato, o PCMCEJTRC é um problema $\mathcal{NP}$-Difícil [5]. Então, é necessário relaxar o problema de forma a permitir ciclos nos caminhos. Este problema é conhecido como problema de caminho mais curto com janela de tempo e restrição de capacidade, ou PCMCJTRC, que também é $\mathcal{NP}$-Difícil, porém pode ser resolvido por programação dinâmica em tempo pseudopolinomial se $t_{ij} > 0$. No PCMCJTRC, um caminho não elementar deve satisfazer as restrições de tempo e capacidade, porém os arcos podem ser utilizados mais de uma vez e os clientes podem aparecer em um mesmo caminho mais de uma vez.

Programação Dinâmica

A programação dinâmica desenvolvida por Desrochers [4] é geralmente utilizada para resolver problemas de caminho mais curto com recursos

limitados. No PCMCJTRC temos dois recursos: tempo e capacidade.

O algoritmo assume que o tempo e a capacidade são valores inteiros e tempos de viagem positivos nos arcos. O algoritmo então discretiza o tempo e a capacidade através do uso de rótulos de três dimensões (i, t, d). Dado um rótulo (i, t, d), i é o último cliente atendido pelo caminho, t determina o tempo em que ele é atendido e d a demanda acumulada deste caminho. O custo associado a este caminho é dado por $c(i, t, d)$. A programação dinâmica inicializa com um rótulo inicial $(0, 0, 0)$ e um custo $c(0, 0, 0)$ igual a zero. A equação de recorrência é dada por:

$$c(i, t, d) = \min_{i}\{\bar{c}_{ij} + c(i\prime, t\prime, d\prime) | t\prime + t_{ij} = t \text{ e } d\prime + d_j = d\}$$

A solução pode ser encontrada através de *backtracking* dos rótulos. Um limite superior do número de rótulos é dado por:

$$\Gamma = \sum_{i \in N}(b_i - a_i)(C - 1)$$

Sabemos que Γ é muito grande, porém em grande parte dos casos, o rótulo não será viável ou então será dominado por algum outro e, portanto não precisa ser considerado. O conceito de dominância será apresentado a seguir. Os rótulos são tratados por ordem crescente de tempo. Quando um rótulo é tratado, ele é expandido para todos nós do grafo de forma que a extensão seja viável e que o custo diminua. O algoritmo inicia com apenas um rótulo e este número aumenta durante a execução do algoritmo, porém este número nunca irá ultrapassar Γ.

O algoritmo 1 descreve o algoritmo de resolução do PCMCJTRC. A função **PróximoRotulo** retorna o rótulo $(i\prime, t\prime, d\prime)$ tal que $i\prime \neq 0$ com o menor $t\prime$ entre todos os rótulos. A função **InsereRotulo** insere um novo rótulo na lista de rótulos ainda não processados (LNNP). Cada vez que um novo rótulo é gerado temos que verificar com os outros rótulos do mesmo vértice se o novo rótulo é dominado por algum outro rótulo ou se ele domina algum.

Algoritmo PCMCJTRC
< Inicialização >
$LNNP = \{(0,0,0)\}$
$c(0,0,0) = 0$
repeat
  Passo 1: Escolhendo o rótulo a ser tratado
  $(i\prime, t\prime, d\prime)$ =PróximoRotulo($LNNP$)
  for *j:=1* ***to*** *n+1* **do**
    Passo 2: Tratamento do rótulo
    if $i \neq j \wedge t\prime + t_{ij} \leq b_j \wedge d\prime + d_j \leq C$ **then**
      < Rótulo Viável >
      if $c(j, \max\{t\prime + t_{ij}, a_j\}, d\prime + d_j) > c(i\prime, t\prime, d\prime) + \hat{c}_{ij}$ **then**
        < Novo Rótulo >
        InsereRotulo(NPS, $(j, \max\{t\prime + t_{ij}, a_j\}, d\prime + d_j)$)
        $c(j, \max\{t\prime + t_{ij}, a_j\}, d\prime + d_j) = c(i\prime, t\prime, d\prime) + \hat{c}_{ij}$
  end
until $i = n + 1$;
return

Algoritmo 1: Algoritmo de Caminho mais curto com janela de tempo e restrição de capacidade

Dominância

O tempo computacional do algoritmo descrito acima pode ser reduzido aplicando o critério de dominância que identifica os rótulos que não precisam ser considerados. Por exemplo, considere os dois rótulos (i, t_1, d_1) e (i, t_2, d_2) tal que:

1. $c(i, t_1, d_1) \leq c(i, t_2, d_2)$
2. $t_1 \leq t_2$
3. $d_1 \leq d_2$

Note que ambos os rótulos correspondem a caminhos que terminam no cliente i e que qualquer extensão viável do segundo rótulo também é viável para o primeiro rótulo. E como o custo do primeiro rótulo é sempre menor ou igual o do segundo, o primeiro rótulo produzirá caminhos tão bons quanto o segundo rótulo. Assim, o segundo rótulo pode ser descartado e as suas possíveis extensões ignoradas.

Definição 4.1 (i, t_1, d_1) ***domina*** (i, t_2, d_2) *se e somente se* $c(i, t_1, d_1) \leq c(i, t_2, d_2)$, $t_1 \leq t_2$ *e* $d_1 \leq d_2$.

No caso das três relações serem iguais, apenas um rótulo precisa ser considerado, mesmo a definição não especificando qual domina o outro. É

possível que nenhum rótulo seja dominado por nenhum outro, porém experimentos computacionais mostraram que na prática vai ocorrer dominância entre rótulos, permitindo uma redução no número de rótulos considerados.

O critério de dominância é aplicado no algoritmo da seguinte forma: Quando um novo rótulo é gerado é preciso saber se este é dominado por algum outro rótulo já existente, ou se o novo rótulo domina algum outro já existente. Assim é necessário comparar o novo rótulo com todos os outros já existentes. Isto pode ser feito armazenando os conjuntos de rótulos em $|N|$ listas encadeadas, uma para cada nó. Com apenas uma percorrida na lista do nó em questão todas as comparações necessárias para o teste de dominância podem ser realizadas.

Eliminação de 2-ciclos

Os caminhos não elementares do espaço de soluções do PCMCJTRC são caminhos que contém ciclos. O 2-ciclo, $[...-i-j-i-...]$, é o ciclo mais comum encontrado em tais caminhos. Para reduzir o número de caminhos não elementares o algoritmo pode ser modificado de forma a eliminar os 2-ciclos. A eliminação do 2-ciclos nos algoritmos de caminho mais curto é um procedimento padrão que foi apresentado pela primeira vez por Houck, Picard, Queyranne e Vemuganti.

Para descrever o novo algoritmo modificado, precisaremos utilizar um novo rótulo de 4 dimensões $(i, t, d, pred)$, onde $pred$ é o nó predecessor ao rótulo, isto é, o nó que precede i no caminho correspondente ao rótulo. Precisaremos também das seguintes definições:

Definição 4.2 *Um rótulo é dito* ***fortemente dominante****, se este não é dominado por nenhum outro rótulo e pelo menos uma das condições a seguir são satisfeitas:*

1. $t + t_{i,pred} > b_{pred}$
2. $d + d_{pred} > C$

O rótulo $(i, t, d, pred)$ é dito ***semi-fortemente dominante*** *se ele não é dominado por nenhum outro e nenhuma das condições acima são satisfeitas.*

O rótulo $(i, t, d, pred)$ é dito ***fracamente dominante*** *se ele é dominado apenas por rótulos fortemente dominantes $(i\prime, t\prime, d\prime, pred\prime)$ com o mesmo predecessor $pred\prime$ e tal que $pred \neq pred\prime$.*

Um rótulo fortemente dominante implica que este não pode ser estendido ao seu predecessor. Um rótulo semi-fortemente dominante implica que o rótulo pode ser extendido ao seu predecessor, porém como estamos querendo eliminar o 2-ciclo, este não é permitido. Ao invés disso, guardamos um rótulo (dominado por um semi-fortemente dominante) com um predecessor diferente e que é possível extendê-lo ao predecessor do rótulo semi-fortemente dominante. Um rótulo fracamente dominante é dominado por outro semi-fortemente dominante e que pode ser extendido apenas para o predecessor deste rótulo.

Para cada estado (i, t, d) pode existir:

1. nenhum rótulo
2. um rótulo fortemente dominante
3. um rótulo semi-fortemente dominante e possibilidade de um rótulo fracamente dominante

Agora, o total de rótulos é limitado por $2.\Gamma$ e a ordem da complexidade computacional não se altera. Como os rótulos não são necessariamente descartados por serem dominados por outros rótulos o segundo passo do algoritmo se torna um pouco mais complicado. Se um novo rótulo é dominado por um rótulo antigo, este pode ser descartado nos seguintes casos:

- se o rótulo antigo é fortemente dominante
- se o rótulo antigo é fracamente dominante
- se o rótulo antigo e o novo têm o mesmo predecessor
- se o rótulo novo não puder se extendido ao predecessor do rótulo antigo
- se o rótulo novo é dominado por dois ou mais rótulos de predecessores diferentes

Rótulos antigos também podem ser descartados ou mudarem seus *status* de dominância para fracamente dominantes, se forem dominados por rótulos novos. As regras para isso podem ser facilmente estabelecidas.

4.3.3 *Branch-and-Bound*

O algoritmo, que utiliza a regra de busca em profundidade na árvore de enumeração, começa com a adição de colunas à formulação *DWM-PRVC* até que não existam mais colunas de custo reduzido negativo, quando então tenta-se adicionar cortes violados. Este esquema é repetido até que os subproblemas de geração de colunas e de separação não possam mais fornecer novos elementos à formulação.

4.4 Instâncias resolvidas

Os experimentos computacionais foram feitos em cima das intâncias propostas por Solomon[3] baseadas nas instâncias geradas por Christofides, Mingozzi e Toth. Para tais instâncias foram adicionadas janelas de tempo e as capacidades dos veículos foram alteradas.

As instâncias são divididas em dois conjuntos. O primeiro conjunto de problemas permite aproximadamente de 5 a 10 clientes por rota devido as restrições de janela e capacidade dos veículos. Já as do segundo conjunto permitem mais de 30 clientes por rota e estas são consideradas as mais difíceis de serem resolvidas.

O primeiro conjunto de instâncias inclui problemas onde os clientes estão localizados aleatoriamente (tipo R), problemas onde os clientes estão localizados em forma de *cluster* (tipo C) e os problemas misturam um pouco de aleatoriedade com a estrutura de *clusters* (tipo RC). Cada problema tem 100 clientes, porém se considerarmos apenas os 25 ou 50 primeiros clientes podemos criar problemas menores. Na versão de 25 ou 50 clientes dos problemas do tipo RC, os clientes estão localizados em forma de *cluster*, porque a estrutura dos *clusters* aparece na primeira metade do problema enquanto que a segunda metade tem a estrutura aleatória.

As coordenadas dos clientes são as mesmas para todos os problemas de um mesmo tipo, mas os problemas diferem em relação à largura das janelas de tempo.

Os cálculos do custo e de tempo de viagem são utilizados por pesquisadores de formas diferentes. Neste trabalho calculamos o custo e o tempo de viagem truncando após uma casa decimal. Seja (x_i, y_i) e (x_j, y_j) as coordenadas dos clientes i e j temos que:

$$c_{ij} = \frac{\lfloor 10\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor}{10}$$

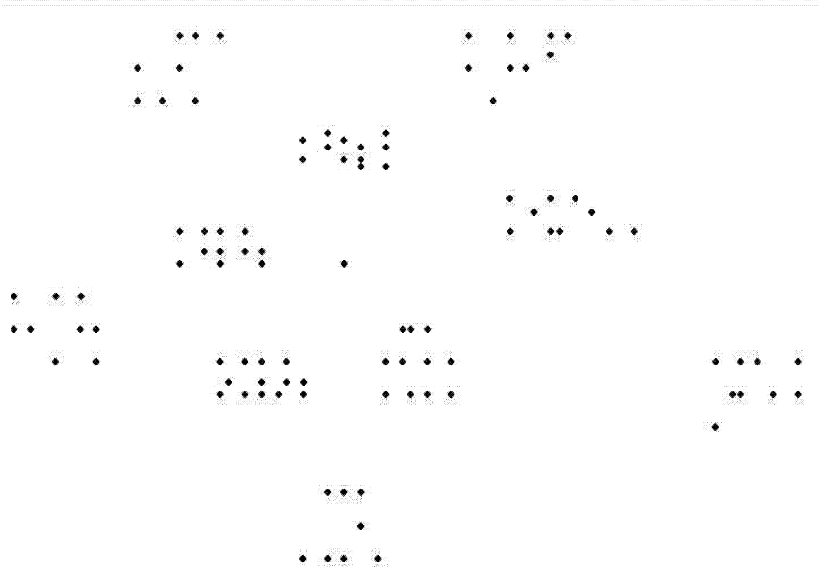

Figura 4.5: Localização dos clientes em instâncias do tipo C.

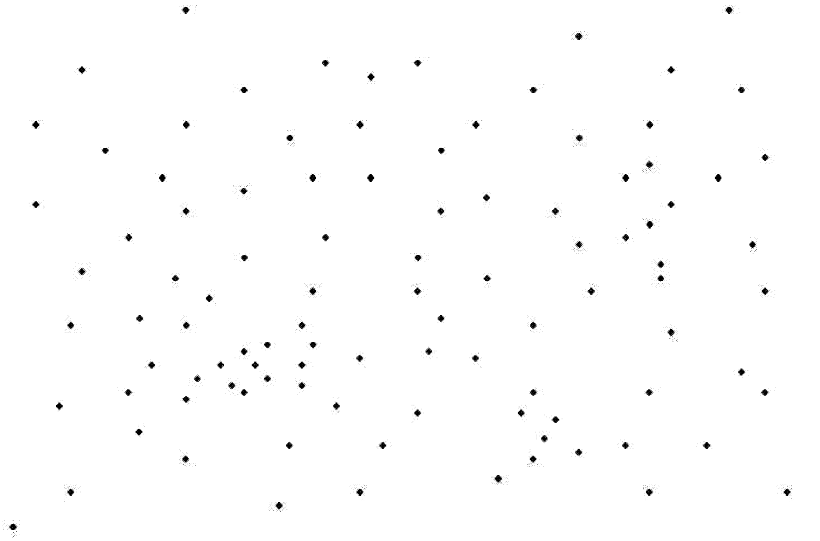

Figura 4.6: Localização dos clientes em instâncias do tipo R.

Os tempos de viagem são definidos como $t_{ij} = c_{ij} + s_i$, onde s_i é o tempo de serviço no cliente i. Assumimos que os custos eram valores inteiros, então multiplicamos c_{ij} por 10 para podermos representa-los de tal forma.

O primeiro conjunto de instâncias é formado por 87 problemas (r101-r112, c101-c109, rc101-rc108), enquanto que o segundo conjunto é formado por 51 problemas (r201-r211, c201-c208, rc201-rc208).

Os experimentos computacionais foram rodados em um PC com processador de 1.8 GHz com 512 MB de memória RAM. Os algoritmos foram implementados em C++ utilizando o Microsoft Visual C++ 6.0. Foi utilizado o resolvedor de programação linear CPLEX 9.0.

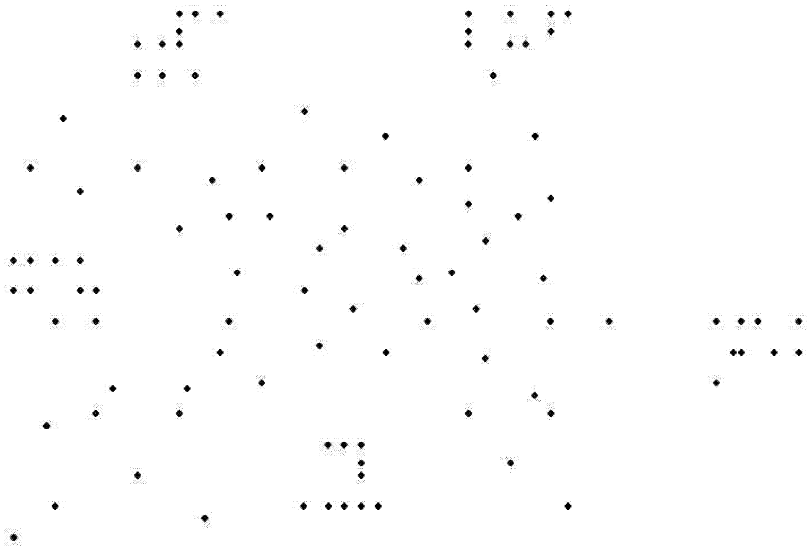

Figura 4.7: Localização dos clientes em instâncias do tipo RC.

4.4.1 Resultados Computacionais

As tabelas 4.1, 4.2, 4.3, 4.4, 4.5 e 4.6 fazem uma comparação de limites inferiores. A primeira coluna indica o tipo da instância e o número de clientes. Por exemplo, a instância r101.25 é uma instância do tipo R1 com 25 clientes. As colunas seguintes LB_1, é o limite inferior encontrado pela geração de colunas sobre as q-rotas, LB_2 é limite inferior obtido resolvendo o nó raiz até a otimalidade, i.e, geração de colunas sobre as q-rotas mais cortes para o PRVC, LB_p é o limite inferior encontrado por Irnich e Villeneuve[47] utilizando geração de colunas e cortes *2-path*, S.O. é a solução do problema e t é o tempo de execução do algoritmo em segundos. Os valores em negrito são valores que se encontram na otimalidade.

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
r101.25	**617,1**	0%	**617,1**	0%	**617,1**	**617,1**	0,03
r101.50	**1044**	0%	**1044**	0%	**1044**	**1044**	0,14
r101.100	1631,15	0,4%	1631,15	0,4%	1634	**1637,7**	205,78
r102.25	546,333	0,14%	546,333	0,14%	**547,1**	**547,1**	3,82
r102.50	**909**	0%	**909**	0%	**909**	**909**	33,02
r102.100	**1466,6**	0%	**1466,6**	0%	**1466,6**	**1466,6**	406,03
r103.25	**454,6**	0%	**454,6**	0%	**454,6**	**454,6**	1,95
r103.50	765,9	0,91%	765,9	0,91%	767,3	**772,9**	397,53
r103.100	1206,31	0,20%	1206,37	0,19%	1206,42	**1208,7**	754,92
r104.25	**461,9**	0%	**461,9**	0%	**461,9**	**461,9**	2,7
r104.50	616,5	1,42%	618,72	1,07%	620,758	**625,4**	972,59
r104.100	949,103	2,31%	950,58	2,15%	951,101	**971,5**	1237,65
r105.25	**530,5**	0%	**530,5**	0%	**530,5**	**530,5**	0,73
r105.50	892,12	0,8%	892,12	0,8%	893,65	**899,3**	3,67
r105.100	1346,14	0,67%	1346,14	0,67%	1349,32	**1355,3**	273,11
r106.25	457,3	1,74%	457,3	1,74%	**465,4**	**465,4**	3,54
r106.50	791,3	0,21%	791,3	0,21%	**793**	**793,0**	20,14
r106.100	1226,44	1,05%	1226,44	1,05%	1227,4	**1239,5**	355,65
r107.25	422,925	0,32%	422,925	0,32%	**424,3**	**424,3**	2,82
r107.50	704,438	0,94%	704,438	0,74%	705,88	**711,1**	74,67
r107.100	1052,13	1,17%	1052,13	1,17%	1052,94	**1064,6**	505,32
r108.25	396,139	0,29%	396,139	0,29%	396,139	**397,3**	2,65
r108.50	588,926	4,66%	588,926	4,66%	595,177	**617,7**	594,01
r108.100	907,21	3,39%	909,94	3,39%	910,617	**939**	1025,44
r109.25	**441,3**	0%	**441,3**	0%	**441,3**	**441,3**	0,95
r109.50	775,096	1,49%	775,096	1,49%	776,231	**786,8**	4,75
r109.100	1130,59	1,42%	1130,60	1,42%	1133,16	**1146,9**	637,83
r110.25	437,3	1,53%	437,3	1,53%	437,363	**444,1**	1,23
r110.50	692,577	0,63%	692,577	0,63%	694,15	**697**	36,5
r110.100	1048,48	1,83%	1048,48	1,83%	1049,94	**1068**	431,73
r111.25	423,788	1,17%	424,0	1,12%	428,05	**428,8**	1,00
r111.50	691,812	2,18%	691,812	2,18%	692,642	**707,2**	62,56
r111.100	1032,03	1,59%	1032,03	1,59%	1032,07	**1048,7**	497,03
r112.25	384,2	2,24%	384,2	2,24%	385,391	**393,0**	1,89
r112.50	607,219	3,65%	607,219	3,65%	612,36	**630,2**	187,87
r112.100	919,192	4,30%	921,77	4,03%	922,412	**960,5**	708,34

Tabela 4.1: Resultados das instâncias do tipo R1

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
r201.25	460,1	0,69%	460,1	0,69%	460,1	**463,3**	3,26
r201.50	788,425	0,44%	791,9	0%	**791,9**	**791,9**	115,00
r201.100	1136,22	0,61%	1138,68	0,4%	1138,65	**1143,2**	5505,23
r202.25	406,35	1,01%	408,35	0,52%	408,35	**410,5**	60,62
r202.50	692,738	0,82%	696,525	0,28%	696,525	**698,5**	630,86
r202.100	-	-	-	-	-	-	-
r203.25	379,88	2,94%	381,625	2,5%	381,625	**391,4**	180,79
r203.50	590,93	2,37%	593,66	1,92%	593,43	**605,3**	636,03
r203.100	-	-	-	-	-	-	-
r204.25	333,075	6,18%	335,35	5,54%	335,35	**355,0**	218,51
r204.50	-	-	-	-	-	-	-
r204.100	-	-	-	-	-	-	-
r205.25	381,283	2,98%	388,45	1,16%	388,45	**393,0**	66,70
r205.50	666,60	3,41%	672,35	2,57%	672,351	**690,1**	715,02
r205.100	-	-	-	-	-	-	-
r206.25	363,132	3,01%	365,908	2,27%	365,908	**374,4**	1313,69
r206.50	605,59	3,64%	611,636	3,28%	611,363	**632,4**	2814,17
r206.100	-	-	-	-	-	-	-
r207.25	347,592	3,87%	349,741	3,28%	349,741	**361,6**	4173,45
r207.50	-	-	-	-	-	-	-
r207.100	-	-	-	-	-	-	-
r208.25	318,105	3,08%	320,038	2,49%	318,922	**328,2**	6248,80
r208.50	-	-	-	-	-	-	-
r208.100	-	-	-	-	-	-	-
r209.25	353,875	4,54%	358,412	3,31%	358,321	**370,7**	1636,61
r209.50	582,877	2,95%	588,413	2,03%	588,413	**600,6**	3380,63
r209.100	-	-	-	-	-	-	-
r210.25	395,844	2,16%	398,06	1,62%	397,906	**404,6**	992,93
r210.50	624,155	3,32%	624,164	3,32%	624,162	**645,6**	1953,55
r210.100	-	-	-	-	-	-	-
r211.25	330,14	5,92%	330,84	5,72%	330,466	**350,9**	5333,08
r211.50	-	-	-	-	-	-	-
r211.100	-	-	-	-	-	-	-

Tabela 4.2: Resultados das instâncias do tipo R2

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
c101.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,21
c101.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	1,02
c101.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	6,44
c102.25	**190,3**	0%	**190,3**	0%	**190,3**	**190,3**	4,26
c102.50	**361,4**	0%	**361,4**	0%	**361,4**	**361,4**	11,40
c102.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	47,70
c103.25	**190,3**	0%	**190,3**	0%	**190,3**	**190,3**	4,56
c103.50	**361,4**	0%	**361,4**	0%	**361,4**	**361,4**	19,45
c103.100	**826,3**	0%	**826,3**	0%	**826,3**	**826,3**	150,52
c104.25	**186,9**	0%	**186,9**	0%	**186,9**	**186,9**	3,15
c104.50	357,25	0,21%	**358**	0%	**358**	**358**	172,34
c104.100	**822,9**	0%	**822,9**	0%	**822,9**	**822,9**	274,21
c105.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,39
c105.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	1,81
c105.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	12,71
c106.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,34
c106.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	1,80
c106.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	18,81
c107.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,45
c107.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	3,39
c107.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	24,23
c108.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,57
c108.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	4,35
c108.100	**827,3**	0%	**827,3**	0%	**827,3**	**827,3**	54,07
c109.25	**191,3**	0%	**191,3**	0%	**191,3**	**191,3**	0,86
c109.50	**362,4**	0%	**362,4**	0%	**362,4**	**362,4**	7,34
c109.100	825,64	0,20%	**827,3**	0%	**827,3**	**827,3**	104,21

Tabela 4.3: Resultados das instâncias do tipo C1

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
c201.25	**214,7**	0%	**214,7**	0%	**214,7**	**214,7**	3,07
c201.50	**360,2**	0%	**360,2**	0%	**360,2**	**360,2**	299,35
c201.100	**589,1**	0%	**589,1**	0%	**589,1**	**589,1**	1000,64
c202.25	**214,7**	0%	**214,7**	0%	**241,7**	**214,7**	20,01
c202.50	**360,2**	0%	**360,2**	0%	**360,2**	**360,2**	625,14
c202.100	-	-	-	-	-	-	-
c203.25	**214,7**	0%	**214,7**	0%	**214,7**	**214,7**	72,57
c203.50	**359,8**	0%	**359,8**	0%	**359,8**	**359,8**	627,23
c203.100	-	-	-	-	-	-	-
c204.25	-	-	-	-	-	-	-
c204.50	-	-	-	-	-	-	-
c204.100	-	-	-	-	-	-	-
c205.25	197,7	7,91%	214,63	0,03%	**214,7**	**214,7**	210,5
c205.50	357,35	0,68%	359,0	0,22%	359,0	**359,8**	758,9
c205.100	-	-	-	-	-	-	-
c206.25	188,621	12,15%	197,232	8,14%	**214,7**	**214,7**	1592,88
c206.50	317,943	11,63%	347,677	3,37%	359,0	**359,8**	8573,09
c206.100	-	-	-	-	-	-	-
c207.25	182,254	15,03%	196,225	8,52%	214,4	**214,5**	984,12
c207.50	-	-	-	-	-	-	-
c207.100	-	-	-	-	-	-	-
c208.25	182,638	14.85%	197,097	8,11%	**214,5**	**214,5**	138,5
c208.50	608,567	11,96%	339,697	3,08%	**350,5**	**350,5**	2483,65
c208.100	-	-	-	-	-	-	-

Tabela 4.4: Resultados das instâncias do tipo C2

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
rc101.25	406,625	11,81%	406,625	11,81%	**461,1**	**461,1**	0,57
rc101.50	850,021	9,96%	850,021	9,96%	**944**	**944**	0,34
rc101.100	1584,09	2,20%	1590,4	1,82%	1617,4	**1619,8**	7,68
rc102.25	**351,8**	0%	**351,8**	0%	**351,8**	**351,8**	0,28
rc102.50	719,902	12,47%	719,902	12,47%	813,037	**822,5**	54,68
rc102.100	1403,65	3,69%	1414,05	2,97%	1439,55	**1457,4**	102,37
rc103.25	332,0	0,24%	332,0	0,24%	332,05	**332,8**	4,68
rc103.50	643,133	9,53%	643,133	9,53%	710,667	**710,9**	14,70
rc103.100	1218,5	3,14%	1235,583	1,78%	1241,72	**1258**	754,26
rc104.25	305,825	0,25%	305,833	0,25%	305,825	**306,6**	15,70
rc104.50	541,8	0,73%	543,75	0,38%	543,75	**545,8**	191,81
rc104.100	1094,34	3,37%	1112,67	1,76%	1112,35	**1132,6**	897,57
rc105.25	410,95	0,09%	410,95	0,09%	410,95	**411,3**	0,87
rc105.50	754,443	11,79%	754,443	11,79%	853,675	**855,3**	2,26
rc105.100	1471,16	2,81%	1472,29	2,74%	1509,8	**1513,7**	43,29
rc106.25	339,24	1,81%	343,2	0,67%	343,2	**345,5**	2,437
rc106.50	585,915	18,98%	588,3	18,65%	717,155	**732,2**	49,24
rc106.100	1086,12	23,76%	1101,01	22,72%	1333,38	**1424,73**	899,00
rc107.25	293,55	1,59%	**298,3**	0%	**298,3**	**298,3**	1,73
rc107.50	597,476	7,04%	597,476	7,04%	632,336	**642,7**	16,96
rc107.100	1170,69	3,07%	1180,84	2,23%	1186,43	**1207,8**	564,0
rc108.25	280,385	4,79%	**294,5**	0%	**294,5**	**294,5**	10,18
rc108.50	538,957	9,89%	538,957	9,89%	590,469	**598,1**	129,64
rc108.100	1063,01	4,59%	1089,36	2,23%	1097,26	**1114,2**	2827,17

Tabela 4.5: Resultados das instâncias do tipo RC1

Instância	LB_1	gap_1	LB_2	gap_2	LB_p	S.O.	t
rc201.25	356,65	0,99%	**360,2**	0%	**360,2**	**360,2**	3,03
rc201.50	670,15	2,13%	681,983	0,41%	681,983	**684,8**	71,32
rc201.100	-	-	-	-	-	-	-
rc202.25	290,408	14,08%	313,389	7,28%	312,992	**338**	871,14
rc202.50	494,358	19,43%	548,405	10,62%	546,359	**613,6**	2379,51
rc202.100	1004,025	8,08%	1013,725	7,19%	1013,69	**1092,3**	3981,25
rc203.25	213,475	34,7%	260,816	20,22%	260,385	**326,9**	1118,61
rc203.50	-	-	-	-	-	-	-
rc203.100	-	-	-	-	-	-	-
rc204.25	188,195	37,21%			244,71	**299,7**	1054,25
rc204.50	-	-	-	-	-	-	-
rc204.100	-	-	-	-	-	-	-
rc205.25	307,54	9,01%	320,787	5,09%	320,787	**338,0**	389,58
rc205.50	334,078	46,99%	408,467	35,18%	581,528	**630,2**	2257,36
rc205.100	-	-	-	-	-	-	-
rc206.25	247,17	23,71%	288,978	10,81%	288,978	**324**	1897,54
rc206.50	330,79	45,77%	407,098	33,26%	532,959	**610**	3774,44
rc206.100	-	-	-	-	-	-	-
rc207.25	216,67	27,37%	288,98	3,12%	263,894	**298,3**	2233,89
rc207.50	390,815	30,01%	467,31	16,34%	469,64	**558,6**	3112,61
rc207.100	-	-	-	-	-	-	-
rc208.25	169,671	37,22%	233,495	13,23%	233,078	**269,1**	1496,57
rc208.50	-	-	-	-	-	-	-
r208.100	-	-	-	-	-	-	-

Tabela 4.6: Resultados das instâncias do tipo RC2