Referências bibliográficas

ANILY, S.H. and G. MOSHEIOV, G. The Traveling Salesman Problem with Delivery and Backhauls, **Operations Research Letters**, 16, 11-18, 1994.

ASSAD, A.A. Modeling and implementation issues in vehicle routing. In: Vehicle Routing: Methods and Studies, B.L. Golden, A.A. Assad (eds), North Holland, Amsterdam, p. 7-46, 1988.

BALLOU, R.H. Logística Empresarial: transportes, administração de materiais e distribuição física. São Paulo: Atlas. 388p, 1993.

BALLOU, R.H. & AGARWALL, Y.K. A performance comparison of several popular algorithms for vehicle routing. **Journal of Business Logistics**, 9, no. 1: 51-65, 1988.

BODIN, L.D.; GOLDEN, B.; ASSAD, A. and BALL, M. Routing and scheduling of vehicles and crews: The state of the art. **Computers and Operations Research**, vol. 10, n. 2, 1983.

CAMPBELL, ANN. The Inventory Routing Problem - The Logistics Institute School of Industrial and Systems Engineering Georgia Institute of Technology.

CASCO, D.O.; GOLDEN, B.L. and WASIL, E.A. Vehicle Routing with Backhauls: Models, Algorithms, and Case Studies, in Vehicle Routing: Methods and Studies, pp. 127-147, B.L, 1988.

CHRISTOFIDES, N.; MINGOZZI, A. AND TOTH, P. The Vehicle Routing Problem, in Combinatorial Optimization, pp. 315-338, N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds.), Wiley, Chichester, 1979.

CLARKE, G. & WRIGHT, J.W. Scheduling of Vehicles from a central depot to a number of delivery points. **Operations Research**, 11: 568-581, 1963.

CONCEIÇÃO, S.V. Impactos da utilização de roteirização de veículos em um centro de distribuição: um estudo de caso. XXIV Encontro Nac. de Eng. de Produção, 2004.

CUNHA, C.B. Uma contribuição para o problema de roteirização de veículos com restrições operacionais. São Paulo: EPUSP, Departamento de Engenharia de Transportes. 222p. (Tese de Doutoramento), 1997.

CUNHA, C.B. Uma Contribuição para o Problema de Roteirização de Veículos com Restrições Operacionais. São Paulo, 1997. 222 p. Tese (Doutorado em Engenharia de Transportes) – Escola Politécnica, Universidade de São Paulo, 1997.

FISHER, M.L. & JAIKUMAR, R. The Local Delivery Problem: Algorithms and Applications. Havard University, Boston, MA 02163, USA, 1984.

FISHER, M.L. and JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. **Networks**, v. 11, p. 109-124, 1981.

GOLDEN, B.L. and ASSAD, A. Vehicle routing: methods and studies. North Holland, Amsterdam, Países Baixos, 1988.

GOLDEN, B.L. Introduction to and Recent Advances in vehicle Routing Mehods. University of Maryland, College Park, Maryland, 1984.

GOLDEN, B.L. & ASSAD, A. Perspectives on Vehicle Routing: Exciting New Developments. University of Maryland, College Park, Maryland, 1986.

HALSE, K. Modeling and Solving Complex Vehicle Routing Problems, PhD thesis, Institute of Mathematical Statistics and Operations Research, Technical University of Denmark, Lyngby, 1992.

LARSON, R.C. & ODONI, A.R. Urban Operations Research. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

LAWLER, E.L.; LENSTRA, J.K.; et all: The Traveling Salesman Problem. Wiley – Interscience Series In Discrete Mathematics And Otimization, 1987.

MATEUS, G.R. O Problema de Roteamento de Veiculos – DCC/UFMG

MOSHEIOV, G. The Travelling Salesman Problem with Pick-up and Delivery. **European Journal of Operational Research**, 79, 299-310, 1994.

NAGY, G. Heuristic Methods for the Many-to-Many Location-Routing Problem, PhD thesis, School of Mathematics and Statistics, The University of Birmingham, Birmingham, 1996.

NOVAES, A.G.N. Sistemas Logísticos: Transporte, Armazenagem e Distribuição de Produtos, Edgard Blucher, São Paulo, 1989.

RONEN, D. Perspectives on practical aspects of truck routing and scheduling. **European Journal of Operational Research**, 35(2):137-145, 1988.

SAVELSBERGH, M.W.P. and SOL, M. The General Pickup and Delivery Problem. **Transportation Science**, 29, 17-29, 1995.

SAVELSBERGH, M.W.P. The General Pickup and Delivery Problem - School of Industrial and Systems Engineering Georgia Institute of Technology, 1995.

SOLOMON, M.M. On the worst-case performance of some heuristics for the vehicle routing and scheduling with time windows constraints. Networks, v. 16, p. 161-174, 1986.

TOTH, P. AND VIGO, D. A Heuristic Algorithm for the Symmetric and Asymmetric Vehicle Routing Problems with Backhauls. **European Journal of Operational Research**, 113, 528-543, 1999.

Anexo I
Modelo IRP

O *Inventory Routing Problem* (IRP) é um problema dinâmico de controle em longo prazo. Este controle em longo prazo é geralmente difícil de formular e também de se resolver. Então todas as abordagens que têm sido propostas e estudadas têm resolvido problemas de planejamentos em curto prazo. Curto prazo significa um único dia, ou pode-se expandir por um período de dias.

A abordagem de curto prazo tem uma tendência a deixar a maior parte dos atendimentos quanto possível para o próximo período de planejamento, o que leva a uma situação indesejável. O planejamento e os objetivos em longo prazo acabam sendo influenciados pelos do curto prazo. O efeito a longo prazo das decisões de curto prazo tem de capturar os custos e os benefícios de entregar a um PCAN mais cedo do que o necessário. Realizando o atendimento mais cedo do que o necessário, o que normalmente significa entregar menos de uma aeronave carregada, pode levar a um aumento dos custos de distribuição futuros, mas reduz o risco e podem, assim, reduzir os custos futuros.

Pode-se distinguir duas abordagens de curto prazo. No primeiro, presume-se que todos os PCANs incluídos na programação de curto prazo têm de ser visitados. Na segunda, assume-se que os PCANs incluídos na programação de curto prazo podem ser visitados. As decisões sobre quem deve ser visitado e quanto deverá ser entregue normalmente são guiadas pelos seguintes suposições sobre o que constituem boas soluções:

- Sempre tentar maximizar a quantidade entregue por visita.
- Sempre tentar enviar aeronaves com a capacidade completa.

Quando o problema de planejamento a curto prazo é de um único dia, o problema pode ser encarado como uma extensão do problema de roteamento de veículos (VRP) e solução e técnicas para o VRP podem ser adaptados. Problemas com um único dia são baseados, normalmente, nas demandas e na previsão para

esse dia. Portanto, ele evita a dificuldade de previsão de longo prazo, o que torna o problema muito simples.

Quando o problema de planejamento a curto prazo consiste de vários dias, o problema torna-se mais complexo, mas tem o potencial de propor soluções muito melhores. Tipicamente, problemas de curto prazo são formulados por programação matemática e resolvidos utilizando técnicas de decomposição, tais como relaxação lagrangeana.

Como já foi mencionado no capítulo 2, o problema IRP se diferencia dos problemas tradicionais de roteamento porque é baseado no consumo de cada cliente ao invés dos seus pedidos. O IRP preocupa-se da distribuição periódica de produtos, tendo como origem um único depósito. No problema existem vários clientes (PCANs) e uma única origem (CECAN) e os produtos a serem entregues são cargas e passageiros. As visitas aos clientes serão feitas por aeronaves do tipo Hérculos C-130. Então, dado um conjunto de n PCANs sobre um horizonte de tempo T e cada PCAN i tem uma taxa demanda ui (volume diário) e tem uma capacidade máxima Ci. A demanda no PCAN i é Ii no tempo 0. Um conjunto de aeronaves m com capacidade Q distribuem as demandas.

O objetivo é minimizar a média dos custos de distribuição durante o período de planejamento. Durante o processo de decisão devem ser feitas as seguintes perguntas: Quando atender aos PCANs ? Qual a quantidade de passageiros e de carga que devem ser distribuídos para um PCAN ? Qual rota de entrega a aeronave deverá usar ?

O problema acima é determinístico e estático devido à premissa que as taxas de demandas são conhecidas e constantes. Obviamente em problemas desse tipo no mundo real, essas demandas são estocásticas e dinâmicas. Então, pode-se sugerir que uma importante variação do IRP está na incerteza das demandas dos PCANs. Esta variação do IRP é conhecida como S*tochastic Inventory Routing Problem (SIRP).*No SIRP são dados a probabilidade da demanda uit de um PCAN i entre os pontos de decisão t e t + 1 para t = 1,..., T – 1. Devido à incerteza da demanda existe sempre uma possibilidade que os PCANs fiquem sem atendimento. E isto nem sempre pode ser evitado. Pode ser usada, como sugestão, uma função de penalidade. Esta função tem componentes fixos e variáveis, isto é, Si+sid, onde Si é o custo fixo do não atendimento, si é o custo variável do não atendimento, e d é a demanda entre o tempo de não atendimento até o

atendimento. O objetivo é escolher a melhor política de atendimento que minimiza o custo médio por unidade de tempo pelo horizonte de planejamento. Para um melhor entendimento dos problemas IRP e SIRP e suas diferenças, serão mostradas nas próximas seções um modelo com um único PCAN e outro modelo envolvendo dois PCANs.

Problema envolvendo um único PCAN

Este problema também pode ser aplicado quando existem vários PCANs, mas sempre um PCAN é visitado somente de cada vez por uma aeronave, e existe um número de aeronaves suficientes para atender a demanda diária de todos os PCANS que existam.

Primeiramente, serão utilizados conceitos do IRP. A taxa de demanda seja u, a capacidade do PCAN C, a necessidade inicial seja I, o custo de entrega seja c, a capacidade da aeronave seja Q, e o horizonte de planejamento seja T. A otimização consiste em evitar que o cada PCAN não seja atendido até que sua capacidade máxima seja atingida. Então, seja vT o custo para um planejamento de um período de extensão T significa:

$$vT = \max(0, [(Tu - I) / (\min(C,Q))])c$$

Agora serão utilizados conceitos do SIRP o qual decidirá se a visita será feita ao PCAN ou não. A demanda U entre pontos de decisão consecutivos, isto é, a demanda diária, é variável com uma distribuição probabilística conhecida.

Jaillet et al. (1997) estudaram a política "dia d" que faz as visitas para cada PCAN todo dia d e atende-o ao máximo possível, a menos que um problema de não atendimento esteja para acontecer. Caso ocorra, uma aeronave é enviada imediatamente, o que ocasiona um custo S. Então, é presumido que o atendimento é imediato, e nenhuma penalidade é imposta. Seja pj a probabilidade de ocorrer um não atendimento no dia j ($1 \leq j \leq d-1$). Então $p = p1 + p2 + \ldots + pd-1$ é a probabilidade que haja um não atendimento e 1-p é a probabilidade que haja atendimentos normais no período [1, ..., d - 1]. Seja, então, vt(d) o custo total esperado para esta política sobre o período T de planejamento. Tem-se para d > T

$$v_T(d) = \sum_{1 \le j \le T} p_j (v_{T-j}(d) + S)$$

E para d ≤ T

$$v_T(d) = \sum_{1 \le j \le d-1} p_j (v_{T-j}(d) + S) + (1-p)(v_{T-d}(d) + c).$$

Como conseqüência, o custo total esperado de atender um PCAN todo dia d em um período de T-dias (T ≥ d) é dado por

$$v_T(d) = \alpha(d) + \beta(d)T + f(T,d)$$

aonde α(d) é uma constante dependendo de d, f(T,d) é uma função que tende a zero a exponencialmente quando T tende ao infinito, e

$$\beta(d) = \frac{pS + (1-p)c}{\sum_{1 \le j \le d} j p_j},$$

com pd = 1 - p. O valor de β(d) é o custo médio por dia. Para encontrar a melhor política será preciso minimizar vT (d), para valores maiores de T, significa encontrar um d para o qual o seu β(d) seja mínimo.

A política do "dia d" tem vantagens que podem ser utilizadas mesmo quando a demanda do PCAN não puder ser medida e mesmo que ela ocorra somente quando estiver ficando sem atendimento. Esta política também possui desvantagens. A primeira é que ela não apresenta soluções ótimas. A segunda é que a probabilidade pj usada na análise do dia d é muito difícil de ser obter e depende da quantidade de vezes de atendimento de cada PCAN.

Problema envolvendo dois PCANs

Quando mais de um PCAN tem que ser atendido, a complexidade do problema aumenta significantemente. Não só é preciso decidir qual o próximo

PCAN que deverá ser visitado, mas também como incluí-lo na rota da aeronave e o quanto de carga/passageiros entregar em cada PCAN. Para problemas deste tipo, existem duas soluções radicais: visitar cada PCAN todo tempo ou visitar os dois juntos. Isto é fácil para expressar o custo associado com cada solução, onde para simplificar assumimos que I1 = I2 = 0:

$$v_T = \max(0, \lceil \frac{T u_1}{\min(C_1, Q)} \rceil) c_1 + \max(0, \lceil \frac{T u_2}{\min(C_2, Q)} \rceil) c_2,$$

onde supondo, implicitamente, que se pode ou implementar a solução com uma aeronave ou que existem duas aeronaves, e

$$v_T = \lceil \frac{T}{\min(\frac{C_1}{u_1}, \frac{C_2}{u_2}, \frac{Q}{u_1+u_2})} \rceil c_{12},$$

onde c12 mostra o custo da rota passando por todos os PCANs.

É fácil imaginar quais destas duas estratégias radicais é a melhor. Entretanto, há outras estratégias possíveis: visitando algumas vezes os PCANs juntos, e visitando-os às vezes separadamente. Será esperado que, quando um PCAN tem uma taxa elevada de demanda ou um menor do que o outro, será visitado diversas vezes, ou ocasionalmente os dois juntos.

Entretanto, se este PCAN não puder receber toda a carga de uma aeronave com a capacidade máxima? Ou, que se os dois estiverem próximos? E, se os visitar juntos quanto da entrega a cada um deles? Sería observado que a resposta não é assim óbvia.

Quando os dois clientes são visitados ao mesmo tempo, está claro que, dado a quantidade entregue no primeiro PCAN, seria ótimo entregar tanto quanto possível no segundo PCAN (determinado pela quantidade restante na aeronave, e pela capacidade restante no segundo PCAN). Assim, o problema de decidir quanto entregar a cada PCAN envolve uma única decisão. Entretanto, esta decisão pode não ser tão fácil, como o seguinte exemplo de SIRP com dois PCANs.

Cada PCAN tem uma capacidade de armazenamento de 20 unidades. As demandas diárias dos PCANs são independentes e distribuídas idênticamente

(através dos PCANs e também através do tempo), com P [demanda = 0] = 0:4 e P [demanda = 10] = 0:6. A penalidade está s1 = 1000 por unidade no PCAN 1 e s2 = 1005 por unidade no PCAN 2. A capacidade da aeronave é 10 unidades. A cada manhã a demanda dos dois PCANs é medida, e o responsável pelas decisões decide quanto entregar a cada um. Há três rotas possíveis para a aeronave realizar: rotas exclusivamente aos PCANs 1 e 2, com custos de 120 cada, e uma rota a ambos os PCANs 1 e 2, com um custo de 180. Somente uma rota pode ser feita por o dia.

Esta situação pode ser modelada como processo de decisão de Markov com um horizonte infinito, com objetivo de minimizar o custo total previsto. Devido ao tamanho reduzido, é possível calcular o valor previsto ótimo e a política ótima. Também é interessante observar que é ótimo entregar mais no PCAN 1 do que no 2, embora a penalidade no PCAN 2 seja maior do que a no PCAN 1, e todos dados restantes, incluindo probabilidades da demanda, custos, são os mesmos para os dois PCANs. Entretanto, esta decisão faz sentido quando observados futuros cenários possíveis. Se no próximo período, o PCAN 1 usar 10 unidades e o PCAN 2 usar unidades 0 (com probabilidade 0,24), então o próximo ponto de decisão das demandas serão 4 e 10 unidades respectivamente, e a aeronave reabastecerá 10 unidades no PCAN 1. Em todos casos restantes (com probabilidade 0,76), a aeronave entregará 10 unidades o PCAN 2 na próxima vez. Assim, em todos os casos, a aeronave visitará somente um PCAN na próxima vez. Também, em todos os casos o PCAN 2 terá 10 ou mais unidades após a entrega na próxima vez, visto que o PCAN 1 terá somente 4 unidades com probabilidade 0,36. Isto ilustra a importância de olhar adiante mais de um período de tempo ao escolher a melhor ação.

Anexo II
Modelo de VRPPD

O modelo escolhido é a heurística para VRPPD misto baseado no artigo de Nagy e Salhi (2003).

Introdução à Heurística Integrada Básica:

Muitos métodos de solução para problemas VRPPD misto são baseados em inserção. Embora esses métodos funcionem bem para um número reduzido de *backhauls*, assim que o número desses *backhauls* começam a aumentar sua complexidade computacional também aumenta proporcionalmente. O mesmo problema surgiria se um método de inserção fosse aplicado a VRPPDs simultâneos (Nagy 1996). Assim, os autores decidiram construir um método que trata tanto os clientes *linehaul* quanto os *backhaul* da mesma forma, em uma heurística integrada. Embora este método tenha sido projetado tendo a princípio VRPPDs simultâneos em mente, funciona bem da mesma forma para o caso misto. Contudo, a metodologia não foi concebida para a hipótese de entregas antes das retiradas, pois este pressuposto é contrário ao tratamento igualitário entre *linehauls e backhauls*.

O veículo usado é a aeronave do tipo Hércules C-130, e a relação existente entre passageiros e cargas em seu compartimento de carga seja dada conforme a tabela abaixo:

PALLETS	PASSAGEIROS
0	80
1	64
2	48
3	32
4	16
5	0

Cada *pallet* pode comportar até 4.500 kg de carga, totalizando 22.500 kg. Cada conjunto de cadeiras acomoda até 16 passageiros, totalizando 80

passageiros. Com base na tabela acima, cada *pallet plt* colocado na aeronave substitui 1 conjunto de cadeiras *cc* e vice-versa. Tem-se então a relação abaixo.

1 plt = 1 cc = 16 passageiros ou 4.500 kg de carga

As rotas das aeronaves são caracterizadas por uma seqüência de PCANs ou de arcos. As rotas são indicadas pelas letras **x**, **y**, **z** e os PCANs por ***a, b***, etc. A oferta e a demanda de carga e de passageiros de um PCAN *a* é indicada por *pp*(*a*), *pc*(*a*) e *qp*(*a*), *qc*(*a*) respectivamente. A oferta e a demanda total de carga e de passageiros de um PCAN *a* é indicada pelas equações:

$$p(a) = pp(a) + pc(a)$$
$$q(a) = qp(a) + qc(a)$$

A capacidade máxima da aeronave é indicada por C, enquanto que a retirada total e a demanda total da rota **x** é indicada por P(**x**) e Q(**x**) respectivamente. As funções são distinguidas por frases em itálico, tais como *load, maxload, minload*: seus significados serão explicados a seguir.

A definição da função de carga para um percurso da aeronave: trata-se simplesmente da carga que a aeronave transporta ao longo dos PCANs (arcos) de uma rota. A carga no arco *ab* é indicada por *load(ab)*. Nagy e Salhi observam que, se o pressuposto de "retiradas após entregas" se aplica, então esta função é sistematicamente decrescente até um determinado ponto e então se torna sistematicamente crescente.

Com isso, desde que a função da carga não esteja acima da capacidade máxima C tanto para o primeiro quanto para o ultimo arco, então a rota é viável. Por outro lado, se o problema cair nas categorias de retiradas e entregas mistas ou simultâneas, então é provável que a função de carga sofra um "pico" em algum lugar no meio do caminho. Em particular, é possível que a função de carga fique abaixo da capacidade máxima para o primeiro e o último arcos, mas não para alguns arcos no caminho. Defina o máximo de carga em uma rota **x**, *maxload*(**x**), como sendo a maior carga total da aeronave durante o percurso. Os autores podem, da mesma forma, definir a carga máxima de uma parte do percurso, *maxload*(*a*,*b*), como sendo o máximo de cargas entre os PCANs *a* e *b* na rota **x**. Uma variável para a menor carga em um percurso da aeronave ou parte do

percurso é também introduzida e denominada *minload*(**x**) e *minload*(*a*,*b*) respectivamente. Se *maxload*(**x**) = C e *minload*(**x**) = 0, então a rota é viável. No algoritmo de solução, a segunda condição será sempre preenchida automaticamente.

Para uma rota com *n* PCANs, todas as funções acima podem ser armazenadas em uma matriz *n* por *n*. A carga para cada arco pode ser armazenada na diagonal frontal da matriz, as cargas mínimas e máximas por setor do trajeto nos triângulos superiores e inferiores respectivamente, e finalmente *maxload* e *minload* ficarão nos cantos inferior esquerdo e superior direito. A função de carga é atualizada juntamente com o percurso, sem nenhum esforço computacional adicional. Para criar ou atualizar a tabela aplica-se um algoritmo cuja complexidade é *O*(*n*). Isso terá que ser feito cada vez que uma rota sofrer mudanças. No entanto, a *maxload* de uma mudança planejada no percurso pode ser calculada em tempo constante, o que torna nosso procedimento livre de esforço, do ponto de vista computacional.

Mosheiov (1994) revelou que qualquer trajeto de um veículo, no caso da aeronave, em deslocamento pode se tornar viável, desde que P(**x**) e Q(**x**) não excedam a capacidade deste. Em seu modelo não houve qualquer restrição na duração máxima do percurso. Este teorema é útil, uma vez que um trajeto inviável de um veículo que mantenha as condições acima pode se tornar viável com a execução de certas operações. Além disso, se tanto a carga total de entrega quanto a carga total de retirada excederem a capacidade da aeronave para cada rota deste trajeto não poderá se tornar viável, mesmo que se faça uma mudança na ordem dos PCANs. Assim, as condições estabelecidas por Mosheiov (1994) são necessárias e, em muitos casos, por exemplo, se a distância restritiva não for muito curta, suficientes para viabilizarem uma rota. Neste estudo, é usado este teorema indiretamente e refere-se a uma rota pela qual nem o total de retiradas, nem o total de entregas excedam a capacidade do veículo e sua duração é inferior à distância máxima, como uma rota de "viabilidade fraca". Se a duração de um percurso não excede a distância máxima e para cada um dos seus arcos a função de carga fica abaixo da capacidade de restrição, denomina-se rota de "viabilidade forte".

A seguir, é explicado como certas mudanças nas rotas alteram suas viabilidades, onde a viabilidade é medida em termos da função *maxload*,

proporcionando assim a base para a transformação do algoritmo VRP num método VRPPD e também para as operações de eliminação das inviabilidades.

Como mudanças nas rotas influenciam a viabilidade

Os procedimentos de melhoria usados na literatura (tais como os de Salhi e Rand, 1993) mudam a estrutura das rotas, muitas vezes pela inserção ou remoção dos PCANs e, por vezes, resultam em uma inversão do sentido em partes do trajeto de uma aeronave.

Enquanto no VRP a direção da rota é irrelevante, para o VRPPD é uma parte essencial do desenho do percurso. Se há muitos PCANs com muita demanda e poucas oferta no começo da rota, esta pode se tornar facilmente inviável. Reverter a rota pode torná-la viável, já que seriam feitas mais ofertas no começo e menos demandas no final. É muito fácil verificar a viabilidade da rota invertida, como demonstra a relação a seguir:

$$maxload(\mathbf{x}') = Q(\mathbf{x}) + P(\mathbf{x}) - minload(\mathbf{x}) \quad (1)$$

onde **x'** se refere à inversão da rota **x**. A relação acima pode ser facilmente verificada de forma algébrica. A função *maxload*(**x'**) é comparada com C para verificar a viabilidade. Também é possível que somente uma parte da rota seja revertida, por exemplo, se for executada uma operação de 2-opt (Lin, 1965). A antiga rota será 0…*ab*…*cd*…0 e a nova 0…*ac*…*bd*…0, sendo 0 o depósito. (Perceba que o procedimento de 2-opt aplicado aos PCANs *a*, *b*, *c* e *d* resulta em uma inversão da rota entre as seções *b* e *c*.) A carga máxima da seção afetada pode ser relacionada à sua carga mínima anterior, como se vê:

$$maxload(c,b) = load(ab) + load(cd) - minload(b,c) \quad (2)$$

Isto é o suficiente para se verificar a viabilidade da nova rota. Outra operação que ocorre com freqüência é a remoção de um PCAN do percurso da aeronave, e sua inserção em algum outro lugar na mesma rota, ou em uma rota diferente. A supressão de um PCAN *a* melhora a viabilidade da rota, uma vez que a cargas e passageiros nos arcos antes de *a* irão diminuir por $q(a)$, e nos arcos

depois de *a* por *p*(*a*). É possível que, com a realocação de um PCAN de uma rota inviável para uma rota de forte viabilidade, a anterior venha a se tornar uma rota também de forte viabilidade, enquanto que a última mantém sua força de viabilidade. A verificação das mudanças de viabilidade em procedimentos baseados em inserção também pode ser feita sem custos computacionais expressivos, usando-se a função *maxload*.

A estrutura da heurística integrada básica

Uma vez que a fraca viabilidade de uma rota VRPPD é muito similar à viabilidade de uma rota VRP, é admitido que uma solução de viabilidade fraca possa ser construída usando-se técnicas padronizadas de VRP. E pode, então, ser transformada em uma solução de forte viabilidade, usando-se rotinas de modificação de rotas.

O método subjacente de roteirização utilizado é o de Salhi e Sari (1997). Este algoritmo baseia-se em uma heurística de construção de rotas para fornecer uma solução inicial e, então, usa uma variedade de rotinas para aperfeiçoar os percursos. Com isso, os autores separam tanto o método que cria a solução inicial quanto o conjunto de rotinas de aperfeiçoamento, de uma só vez. Rotinas de aperfeiçoamento podem ser modificadas para melhorar a viabilidade das rotas. Por ultimo, uma vez que a forte viabilidade é alcançada, ainda é possível aperfeiçoar as rotas, desde que a viabilidade forte seja mantida.

A heurística integrada (IH), consiste nas seguintes quatro fases:

- Fase 1: Encontrar uma solução inicial de fraca viabilidade
- Fase 2: Aperfeiçoar esta solução mantendo a viabilidade fraca
- Fase 3: Fortalecer a viabilidade da solução
- Fase 4: Aperfeiçoar a solução mantendo a viabilidade forte

A heurística de construção de rotas (fase 1)

Este algoritmo é baseado no conceito de "múltiplas viagens longas", conforme exposto por Salhi, Sari, Saidi e Touati (1992). Os autores devem olhar

ao mesmo tempo para o algoritmo original de roteirização da aeronave e para a forma como foi modificada a rota, visando corrigir o problema das retiradas e entregas. Na versão VRP, é primeiramente criado um trajeto longo, incluindo o CECAN. Este é um trajeto que começa no CECAN, atravessa todos PCANs e retorna ao CECAN. Em seguida, um custo de rede é criado. Este é um gráfico dirigido, onde os PCANs são os pontos e as bordas são definidas conforme se segue. Definindo o trajeto Tab como um trajeto que se inicia com um arco que vai do CECAN ao PCAN *a*, na seqüência faz o trajeto longo entre os PCANs *a* e *b* e termina com um arco do PCAN *b* até o CECAN. Então, existe uma borda dirigida no custo de rede de *a* até *b* se, e somente se, o trajeto Tab for viável em termos de capacidade da aeronave e restrição de distância. O comprimento da borda *ab* no custo de rede é o comprimento de Tab. Em terceiro lugar, o problema do caminho menor é resolvido usando-se o algoritmo de Dijkstra (1959), que dá um particionamento do trajeto mais longo. Este procedimento é repetido a partir de diferentes trajetos longos e é escolhida a solução de menor custo global. Nos experimentos são construídos 5 trajetos gigantes; um utilizando o vizinho mais próximo, outro a menor regra de custo de inserção e os três trajetos restantes são gerados de forma aleatória. Uma descrição detalhada de como gerar esses trajetos gigantes e como construir as redes de custo associadas pode ser encontrada em Salhi *et al.* (1992).

É o particionamento acima do trajeto gigante, em particular a criação da rede de custo, que precisa ser modificada para o caso das retiradas e entregas, uma vez que a viabilidade deve ser verificada em termos de uma função de carga variável. Para o VRP, a viabilidade da rota é verificada através da comparação da demanda total da rota Q(**x**) versus a capacidade máxima C. Para o VRPPD, os autores têm que introduzir a oferta total P(**x**) nesta rotina e aceitar como sendo uma rota de fraca viabilidade se a desigualdade

$$\max(Q(\mathbf{x}),P(\mathbf{x})) = C \quad (3)$$

estiver satisfeita. É observado que a extensão deste algoritmo original não aumenta a complexidade computacional. Outras partes deste método também precisam ser alteradas em conformidade: tais mudanças são mínimas e também não aumentam a complexidade computacional. A complexidade global desta fase

é $O(n2)$, em que n é o número total de PCANs (Salhi *et al.*, 1992). Embora as mudanças sejam suficientes para atingir a viabilidade fraca, elas não garantem uma forte viabilidade. Os autores apresentam uma versão do método que pode produzir soluções iniciais de forte viabilidade na subseção (5.1.). Finalmente, ao final desta fase, é dada às rotas a direção que oferece a menor possibilidade de inviabilidade.

Aperfeiçoando/Viabilizando rotinas

Os autores pegaram as rotinas de aperfeiçoamento do VRP de Salhi e Rand (1993) e as modificaram para o caso do VRPPD. Um número de rotinas novas também foram adicionadas quando necessário. A maior parte delas pode ser usada tanto para aperfeiçoamento, reduzindo a distância total, quanto para reduzir o percentual de inviabilidade (definido como *maxload*–C para cada rota). Algumas dessas rotinas são explicadas de forma mais detalhada do que outras, dependendo de sua importância. Os autores remetem o leitor para Salhi e Rand (1993) para uma explicação mais detalhada das versões do VRP para algumas das rotinas, incluindo ilustrações gráficas. A subseção é concluída com uma breve discussão a respeito da complexidade computacional das rotinas.

Rotina REVERSA. Esta é uma nova rotina, baseada na observação de que, ao reverter o sentido de uma rota, a viabilidade pode ser melhorada sem que haja aumento de tamanho. O procedimento simplesmente escolhe a direção da rota que tiver o menor *maxload.*

Rotina 2-OPT. Este método, primeiramente descrito por Lin (1965), é baseado em arcos intercambiáveis, digamos *ab* e *cd* com *ac* e *bd*. O sentido da rota será revertido entre os PCANs *b* e *c*. Os autores supõem que inviabilidades no VRPPD ocorrem em virtude do fato de que os PCANs estarem "na ordem errada" no percurso da aeronave. Assim, é razoável que se reorganize a ordem dos PCANs em uma rota pela reversão do sentido nas partes da rota em que as inviabilidades ocorrerem. Os autores observam que a rota retorna ao seu estado original se esta mudança for aplicada duas vezes, assim 2-OPT pode ser visto como a sua própria dupla. Pode ser usado tanto para reduzir a distância da rota

quanto para reduzir a ocorrência de inviabilidades. Esta observação é verdadeira para a maior parte das rotinas subseqüentes.

Rotina 3-OPT. Esta é uma ligeira modificação da rotina original de Lin (1965), que se baseia na troca de quaisquer três arcos com outros três arcos. O método modificado considera somente três arcos consecutivos e, justamente por isso, é muito mais rápido.

Rotina SHIFT. Esta rotina envolve duas rotas, mas ainda assim é muito similar à versão do 3-OPT. Envolve a supressão de um PCAN de uma rota e a inserção deste em um arco de outra rota.

Rotina EXCHANGE. Esta rotina é uma extensão à SHIFT, em que dois PCANs são simultaneamente inseridos em cada uma das antigas rotas destes, mas não necessariamente nas localizações anteriores de cada um.

Rotina PERTURB. Esta é outra modificação da SHIFT, que considera três rotas de uma vez. Um PCAN *a* é removido da rota x e inserido na rota y, ao mesmo tempo em que um PCAN *b* é removido da rota y e inserido em uma terceira rota z. Esta operação difere de se aplicar SHIFT duas vezes, já que após a primeira operação SHIFT pode encontrar em uma situação inviável (ou seja, nem sequer de fraca viabilidade). Além disso, a complexidade desta operação é maior, pois há mais possibilidades a serem consideradas dentro de um mesmo movimento.

Rotina REINSERT. Este método se origina de um teorema de Mosheiov (1994). O autor provou que uma rota de fraca viabilidade de um vendedor em deslocamento com retiradas e entregas pode sempre se tornar fortemente viável através da reinseração do CECAN. Sendo assim, a rotina REINSERT considera todos os arcos em um tour para possível reinserção do depósito, conforme a Figura 2. Mosheiov (1994) demonstrou como esta rotina pode ser útil na eliminação de inviabilidades, entretanto é também possível que voltar a introduzir o depósito possa acabar por reduzir o comprimento do percurso.

Rotinas NECK e UNNECK. Trata-se de um novo conjunto de rotinas concebidas especialmente para o VRPPD. De fato, elas são aplicáveis apenas no problema das rotinas e entregas simultâneas e somente se os PCANs estiverem autorizados a serem visitados duas vezes. É possível que tenhamos PCANs próximos ao depósito que possuam grandes demandas ou níveis de oferta. Embora REVERSE procure garantir que todos os PCANs com mais ofertas sejam postos no final, e aqueles com mais demandas fiquem no início do percurso da aeronave, pode ser que ainda hajam alguns PCANs causando inviabilidades, principalmente se tanto a demanda quanto a oferta forem altos. No entanto, pode-se servir a estes PCANs duas vezes: primeiro para a entrega e depois para a retirada. Assim, NECK divide um PCAN *a* em duas entidades: um PCAN de retirada *ap* e um PCAN de entrega *ad*. A entidade de entrega é deixada onde estiver e a de retirada é considerada para inserção dentro de todos os arcos da rota seguinte à entrega, ou vice versa. Os autores salientam que *maxload*(*ad ap*) é reduzido tanto por $p(a)$ quanto por $q(a)$. Diferentemente das rotinas anteriores, NECK não é sua própria dupla, e para isso é introduzida uma rotina denominada UNNECK, que é a dupla de NECK e busca unir as entidades separadas a e a, se tal opção for considerada viável. Assim sendo, NECK é a rotina viável enquanto que UNNECK é a rotina de aperfeiçoamento.

Rotinas IDLE e CREATE. Rotina IDLE é simplesmente uma rotina de serviço concebida para remover quaisquer rotas que não contenham nenhum PCAN e não representem nenhum movimento físico da aeronave. É incluída no grupo de rotinas de aperfeiçoamento apresentadas por Salhi e Rand (1993). É construída uma dupla para IDLE chamada CREATE, que cria uma rota vazia. Isto, por si só, não afeta nem a otimalidade nem a viabilidade, mas pode vir a ser útil, já que mais tarde algumas rotinas podem mover um PCAN dentro desta rota 'fictícia' e, assim, criar outra rota viável.

Rotinas COMBINE e SPLIT. A rotina COMBINE tenta combinar duas rotas, caso isto seja viável. Os autores observam que há várias formas de se juntar duas rotas, principalmente se a direção da rota for importante (não era para a versão do VRP original mas é para o VRPPD). O *maxload* de uma rota combinada é simplesmente o somatório dos valores de *maxload* das duas rotas originais. Os

autores criaram uma dupla para COMBINE denominada SPLIT, que pode aperfeiçoar a viabilidade. Excolhe uma rota e encontra-se o melhor arco (digamos *ab*), onde ela pode ser dividida conforme a Figura 4.

Complexidade computacional. A complexidade computacional das rotinas acima é a seguinte. REVERSE: $O(n)$, 2-OPT: $O(n2)$, 3-OPT(modified): $O(n2)$, SHIFT: $O(n2)$, EXCHANGE: $O(n4)$, PERTURB: $O(n3)$, REINSERT: $O(n)$, NECK/UNNECK: $O(n2)$, IDLE/CREATE: $O(n)$, COMBINE/SPLIT: $O(n)$, onde n é o número total de PCANs. (Nagy 1996 e Salhi e Rand 1993).

O aperfeiçoamento composto/fases de viabilidade (fases 2, 3 e 4)

As rotinas de aperfeiçoamento/viabilidade apresentadas na subseção anterior podem ser usadas em três diferentes contextos. Podem tanto ser usadas para melhorar a otimização enquanto mantêm viabilidade fraca ou forte ou podem ser usadas para aperfeiçoar a viabilidade. As estruturas das três fases estão descritas abaixo.

Descrição da fase 2. É usada para promover uma melhora na qualidade da solução, mantendo ao mesmo tempo uma fraca viabilidade. Os módulos são ordenados na seguinte seqüência:

1. 2-OPT, 3-OPT, REINSERT, REVERSE
2. COMBINE, 2-OPT, 3-OPT, REINSERT, REVERSE
3. EXCHANGE, IDLE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE
4. SHIFT, IDLE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE
5. PERTURB, IDLE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE
6. EXCHANGE, IDLE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE
7. COMBINE, IDLE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE

O ordenamento dos módulos reflete seu esforço computacional. Módulos rápidos (2-OPT, 3-OPT, REINSERT e IDLE) são ordenados com mais freqüência, alternando com módulos mais lentos. Estes módulos lentos são ordenados primeiro em uma ordem crescente, e depois em uma ordem decrescente

de complexidade. Este ordenamento é algo arbitrário e qualquer planejamento adequado pode valer a pena ser tentado. Embora REVERSE não seja uma rotina de aperfeiçoamento, não afeta o comprimento da rota e reduz o volume de inviabilidades apresentadas já nesta fase.

Descrição da fase 3. O objetivo desta fase é reduzir o volume de inviabilidade. O processo se inicia com uma solução de viabilidade fraca e o resultado é uma solução de viabilidade forte. A seqüência de módulos é apresentada abaixo.

1. 2-OPT, 3-OPT, REINSERT, REVERSE, NECK, REVERSE,
2. EXCHANGE, REVERSE, 2-OPT, 3-OPT, REINSERT, REVERSE, SHIFT, REVERSE
3. 2-OPT, 3-OPT, REINSERT, REVERSE, PERTURB, REVERSE
4. 2-OPT, 3-OPT, REINSERT, REVERSE, NECK, REVERSE
5. SPLIT, REVERSE, CREATE, REVERSE

Como SPLIT e CREATE geram ambas as rotas adicionais, os autores as usam ao final da heurística composta para eliminar quaisquer inviabilidades que tenham sobrado. Se houver falha nos estágios prévios em encontrar uma solução fortemente viável, então o procedimento é repetido uma vez mais para permitir que as rotas sejam reorganizadas. Se o problema for misto, ou os PCANs só puderem ser visitados uma única vez, o módulo NECK não deve ser aplicado. Nestes casos, NECK e REVERSE são retirados do final das etapas 1 e 4 acima.

Descrição da fase 4. A fase 3 pode implicar um aumento no custo de roteirização e é por isso que os autores empregam uma outra fase de aperfeiçoamento. A fase 4 tem por alvo melhorar a qualidade da solução, enquanto mantém uma forte viabilidade. Sua estrutura é praticamente idêntica à da fase 2, exceto que um ordenamento de UNNECK e REVERSE é adicionado no início das etapas 2 e 7 se o modulo NECK tiver sido usado na fase 3. No entanto, é mais lento do que a fase 2, e analisar a viabilidade forte leva mais tempo do que verificar apenas a fraca viabilidade.

Complexidade computacional. As rotinas acima devem atualizar os valores de *maxload* e *minload*. O processo de atualização tem complexidade $O(n^2)$, confira em Nagy (1996), p.157. Assim, a complexidade geral das fases 2, 3 e 4 – e conseqüentemente de todo o algoritmo – é $O(n^4)$.

Versão usando fator de penalidade

A heurística de construção de rotas apresentada produz uma solução de fraca viabilidade inicial. É possível modificar este método para produzir uma solução inicial de forte viabilidade. No entanto, na prática isto geralmente produz uma solução de baixa qualidade, em que até mesmo os procedimentos de aperfeiçoamento podem não corrigir o problema de forma suficiente. Em virtude disto, um fator de penalidade é introduzido neste método para reduzir a quantidade de inviabilidades na solução inicial.

Na heurística de construção de rotas, um trajeto é viável em relação ao custo de rede se (2) se mantiver. Para conseguir uma viabilidade forte, ao invés de uma mera viabilidade fraca, *maxload*(**x**) deve ser menor ou igual a C. (Com uma ligeira variação para levar em conta a possível reversão da rota.) Como os autores podem não atingir a viabilidade forte, é introduzido um fator de penalidade **?** e utilizado o critério abaixo.

$$\mathbf{?} \cdot maxload(\mathbf{x}) + (1-?) \cdot \max(P(\mathbf{x}),Q(\mathbf{x})) = C \ (0=?=l) \ (4)$$

Quanto maior o valor de **?** se tornar, mais a solução inicial será empurrada em direção a uma viabilidade forte.

Nagy e Salhi apresentam abaixo um pseudo-código na nova heurística composta, que é denominada pela abreviação PEN:

1. Encontrar uma solução inicial de fraca viabilidade, onde as rotas são aceitas caso (3) seja satisfeito.
2. Aperfeiçoar esta solução enquanto se mantém a viabilidade fraca
3. Tornar a solução forte em viabilidade

4. Aperfeiçoar a solução enquanto se mantém a forte viabilidade

Uma versão alternativa

Esta versão se apóia na idéia de alternância entre as rotinas de aperfeiçoamento e de viabilidade e, conseqüentemente, entre soluções viáveis e inviáveis. (Perceba que a viabilidade e a inviabilidade daqui em diante se refere ao limite máximo de carga; o limite máximo de distância jamais é violado. A idéia de se atravessar regiões inviáveis tem a flexibilidade de se poder explorar boas regiões viáveis que não seriam visitadas de outra forma, confira em Kelly, Golden e Assad (1993) para mais detalhes nesse assunto.

Para esta versão alternativa, são desenvolvidas todas as rotinas de aperfeiçoamento. Elas permitem violações na viabilidade forte, mas as controlam usando o parâmetro d. Isto é denominado de *total de violação de viabilidade permitida* e é definido por $\sum i \max(maxload(\mathbf{i})–C;0)$. Este processo flexível envolve a mudança de todos os procedimentos de aperfeiçoamento, a fim de que permitam algumas violações ao verificar a viabilidade, contanto que permaneçam em valores abaixo da mudança na porcentagem calculada a partir de d e do comprimento da rota.

Os autores apresentam abaixo o pseudo-código do método alternativo, referindo-se a ele como procedimento ALT:

1. Estabelecer uma solução inicial de fraca viabilidade
2. Aperfeiçoar esta solução enquanto se mantém a viabilidade fraca
3. Calcular o total de violação de viabilidade dold
4. Tornar a solução fortemente viável
5. Aperfeiçoar a solução com a permissão de uma máxima violação de viabilidade de (dold/2)
6. Fazer dold = dnew e calcular o novo total de violação de viabilidade dnew
7. Se a solução for a mesma do procedimento anterior então parar; do contrário repetir etapas 4 a 7

Pode ser facilmente verificado que este método irá sempre se encerrar num período finito.

Uma versão similar à acima, que usa oscilação estratégica, também foi desenvolvida. A versão usa um cronograma de oscilação, conforme proposto por Kelly, Golden e Assad (1993) e amplia a abordagem de ALT através da fusão das rotinas de aperfeiçoamento e viabilidade em rotinas compostas, que tentam minimizar uma pesada combinação de comprimento de rota e quantidade de inviabilidades. Os autores não fornecem uma descrição desta versão, já que os resultados obtidos não foram suficientemente diferentes daqueles encontrados com ALT. Os autores remetem o leitor a Nagy (1996) para maiores detalhes.

Notação

L é conjunto de CECAN, L = {1, 2, ... , t} L=1

H é conjunto de PCANs, H = {1, 2, ... , n}

K é conjunto de aeronaves, K = {1, 2, ... , m}

dij é distância entre o PCAN *i e j*

qi é demanda total do PCAN *i*

pi é oferta total do PCAN *i*

qpi é demanda de passageiros do PCAN *i*

ppi é oferta de passageiros do PCAN *i*

qci é demanda de carga do PCAN *i*

pci é oferta de carga do PCAN *i*

D é a distância máxima que cada aeronave pode percorrer um uma rota

C é a capacidade máxima da aeronave

xijk = 1, se o arco *ij* é parte da rota *k* 0, se forem outros

tijk a carga no arco *ij* da rota *k* da aeronave

Formulação

Minimizar $\sum k \in K \sum i \in H \cup L \sum j \in H \cup L$ dij · xijk

Sujeito a

$\sum i \in H \cup L \sum j \in H \cup L$ dij · xijk ≤ D ($k \in K$)

(A.1.)

$$t_{ijk} = C \quad (i,j \in H \cup L,\ k \in K)$$

(A.2.)

$$\sum_{k \in K} \sum_{i \in H \cup L} x_{ijk} = 1 \quad (j \in H)$$

(A.3.)

$$\sum_{k \in K} \sum_{i \in H \cup L} x_{ijk} \geq 1 \quad (j \in L)$$

(A.4.)

$$\sum_{k \in K} \sum_{i \in S} \sum_{j \in (H \cup L) \setminus S} x_{ijk} \geq 1 \quad (2 \leq |S|)$$

(A.5.)

$$\sum_{i \in H \cup L} x_{ijk} = \sum_{i \in H \cup L} x_{jik} \quad (j \in H \cup L,\ k \in K)$$

(A.6.)

$$\sum_{i \in H \cup L} \sum_{j \in H \cup L} x_{ijk} \leq 1 \quad (k \in K)$$

(A.7.)

$$t_{ijk} \leq x_{ijk} \cdot \sum_{h} (q_h + p_h) \quad (i,j \in H \cup L,\ k \in K)$$

(A.8.)

$$\sum_{k \in K} \sum_{i \in H \cup L} t_{ijk} - q_j = \sum_{k \in K} \sum_{i \in H \cup L} t_{jik} - r_j \quad (j \in H)$$

(A.9.)

$$x_{ijk} = 0, 1 \quad (i,j \in H \cup L,\ k \in K)$$

(A.10.)

$$t_{ijk} \geq 0 \quad (i,j \in H \cup L,\ k \in K)$$

(A.11.)

Anexo III

Código-Fonte do Programa em Pascal de Roteirização de Veículos com Restrição de Tempo e Capacidade usando o Método Clarke-Wright

```
PROGRAM CAP10EX6;

(* ROTEIRIZACAO DE VEICULOS COM RESTRICAO DE TEMPO E
CAPACIDADE
USADO O METODO CLARKE-WRIGHT
ANTONIO G. N. NOVAES - SET 1988 *)

USES CRT;

CONST
LINHA='----------';
TIT1='ROTEIRIZACAO DE VEICULOS';
TIT2='CAP10EX6';

VAR
N,M,MI,MN,I,J,L,K,K1,NK,NV,NVA,I0,J0,FLAGI,FLAGJ,XI,XJ,IS,JS,NVI,NVJ,CSI,C
SJ : INTEGER;
NVT,VA,FATIV : ARRAY [1..40] OF INTEGER;
VT : ARRAY [1..40,1..40] OF INTEGER;
IA,JA : ARRAY [1..800] OF INTEGER;
X,Y,T1,TX,WX,W,DD : ARRAY [1..40] OF REAL;
TAL : ARRAY [1..40,1..40] OF REAL;
S : ARRAY [1..800] OF REAL;
STATUS,OK1,OK2 : BOOLEAN;
LC : CHAR;
LIN : STRING[80];
TP,V,K0,XD,YD,TT,TC,WV,WT,D1,D2,D3,S0 : REAL;

PROCEDURE TELA;
(* FORMATACAO DA TELA *)
BEGIN
        CLRSCR;
        GOTOXY(1,1);WRITELN(TIT1);
        GOTOXY(70,1);WRITELN(TIT2);
        WRITELN(LIN);
END;

PROCEDURE ENTER_1;
(* ENTRADA DOS DADOS *)
BEGIN
        STATUS:=FALSE;
        WHILE NOT STATUS DO
        BEGIN
                TELA;
```

```
                GOTOXY(10,4);WRITE('> NUMERO DE PONTOS DE PARADA
(EXCETO DEPOSITO):');
                GOTOXY(70,4);READLN(N);
                GOTOXY(10,6);WRITE('> TEMPO MEDIO DE PARADA
(HORAS):');
                GOTOXY(70,6);READLN(TP);
                GOTOXY(10,8);WRITE('> VELOCIDADE MEDIA (KM/H):');
                GOTOXY(70,8);READLN(V);
                GOTOXY(10,10);WRITE('> FATOR DE CORRECAO DE
DISTANCIA:');
                GOTOXY(70,10);READLN(K0);
                GOTOXY(10,12);WRITE('> TEMPO DE CICLO (HORAS):');
                GOTOXY(70,12);READLN(TC);
                GOTOXY(10,14);WRITE('> CAPACIDADE DO VEICULO (TON):');
                GOTOXY(70,14);READLN(WV);
                GOTOXY(10,16);WRITE('> COORDENADAS DO DEPOSITO:');
                GOTOXY(10,18);WRITE('> VALOR DE X (KM):');
                GOTOXY(10,20);WRITE('> VALOR DE Y (KM):');
                GOTOXY(70,18);READLN(XD);
                GOTOXY(70,20);READLN(YD);
                GOTOXY(1,23);WRITELN(LIN);
                GOTOXY(10,24);WRITE('DADOS CORRETOS (S/N)?');
                GOTOXY(70,24);READLN(LC);LC:=UPCASE(LC);
                IF (LC='S') THEN STATUS:=TRUE;
        END;
END;

PROCEDURE ENTER_2;
(* ENTRADA DE COORDENADAS DOS PONTOS *)
BEGIN
        FOR I:= 1 TO N DO
        BEGIN
                STATUS:=FALSE;
                WHILE NOT STATUS DO
                BEGIN
                        TELA;
                        GOTOXY(10,4);WRITE('* * COORDENADA DO PONTO',I:3,'
* *');
                        GOTOXY(10,7);WRITE('> VALOR DE X (KM):');
                        GOTOXY(10,9);WRITE('> VALOR DE Y (KM):');
                        GOTOXY(70,7);READLN(X[I]);
                        GOTOXY(70,9);READLN(Y[I]);
                        GOTOXY(10,12);WRITE(' * * PESO MEDIO DA CARGA
(TON):');
                        GOTOXY(10,14);WRITE('> VALOR (TON):');
                        GOTOXY(70,14);READLN(W[I]);
                        GOTOXY(1,23);WRITELN(LIN);
                        GOTOXY(10,24);WRITE('DADOS CORRETOS (S/N)?');
                        GOTOXY(70,24);READLN(LC);LC:=UPCASE(LC);
                        IF (LC='S') THEN STATUS:=TRUE;
                END;
        END;
END;

PROCEDURE DADOS;
```

```
(* DADOS PARA DEMONSTRACAO EX 10.6 *)
BEGIN

(*
Roteiro 1:

N:=5;TP:=2;V:=700.0;K0:=1.35;WV:=22.0;
X[1]:=240;Y[1]:=20;
X[2]:=385;Y[2]:=55;
X[3]:=1100;Y[3]:=715;
X[4]:=550;Y[4]:=480;
X[5]:=440;Y[5]:=135;
W[1]:=2000;W[2]:= 650;W[3]:= 470;W[4]:= 940; W[5]:= 3240;
XD:=0.0;YD:=0.0; TC:= 48.0;

Roteiro 2:
N:=7;TP:=2;V:=700.0;K0:=1.35;WV:=22.0;
X[1]:=40;Y[1]:=385;
X[4]:=440;Y[4]:=785;
X[3]:=1855;Y[3]:=2200;
X[2]:=460;Y[2]:=2365;
X[5]:=990;Y[5]:=1630;
X[6]:=365;Y[6]:=120;
X[7]:=385;Y[7]:=55;
W[1]:= 355;W[2]:= 2345;W[3]:= 6543;W[4]:= 634; W[5]:= 234; W[6]:= 3894; W[7]:=
4564;
XD:=0.0;YD:=0.0; TC:= 96.0;

Roteiro 3:
N:=8;TP:=1;V:=700.0;K0:=1.35;WV:=22.0;
X[1]:=555;Y[1]:=1080;
X[2]:=990;Y[2]:=1630;
X[3]:=1870;Y[3]:=660;
X[4]:=460;Y[4]:=2365;
X[5]:=20;Y[5]:=2285;
X[6]:=1820;Y[6]:=2950;
X[7]:=1100;Y[7]:=715;
X[8]:=550;Y[8]:=755;
W[1]:=394;W[2]:=3453;W[3]:=2332;W[4]:=1231;
W[5]:=480;W[6]:= 987;W[7]:= 1020;W[8]:= 1890;
XD:=0.0;YD:=0.0;TC:= 72.0;

Roteiro 4
N:=20;TP:=2;V:=700.0;K0:=1.35;WV:=22.0;

X[1]:=555;Y[1]:=1080;
X[2]:=990;Y[2]:=1630;
X[3]:=1870;Y[3]:=660;
X[4]:=460;Y[4]:=2365;
X[5]:=20;Y[5]:=2285;
X[6]:=1820;Y[6]:=2950;
X[7]:=1100;Y[7]:=715;
X[8]:=550;Y[8]:=755;

X[9]:=40;Y[9]:=385;
```

```
X[10]:=440;Y[10]:=785;
X[11]:=1855;Y[11]:=2200;
X[12]:=460;Y[12]:=2365;
X[13]:=990;Y[13]:=1630;
X[14]:=365;Y[14]:=120;
X[15]:=385;Y[15]:=55;

X[16]:=240;Y[16]:=20;
X[17]:=385;Y[17]:=55;
X[18]:=1100;Y[18]:=715;
X[19]:=550;Y[19]:=480;
X[20]:=440;Y[20]:=135;

W[1]:=394;W[2]:=3453;W[3]:=2332;W[4]:=1231;W[5]:=480;W[6]:=987;W[7]:=
1020;W[8]:= 1890;
W[9]:= 355;W[10]:= 2345;W[11]:= 6543;W[12]:= 634; W[13]:= 234; W[14]:= 3894;
W[15]:= 4564;
W[16]:=2000;W[17]:= 650;W[18]:= 470;W[19]:= 940; W[20]:= 3240;

XD:=0.0;YD:=0.0;TC:=72.0;

*)
        (* MUDANCA DE ESCALA *)
        FOR I:= 1 TO N DO
        BEGIN
                X[I]:=X[I]/5+5;
                Y[I]:=Y[I]/5+10;
                W[I]:=W[I]*0.001;
        END;
END;

PROCEDURE PARAMETROS;
(* CALCULA DISTANCIAS, TEMPO E GANHOS *)
BEGIN
        FOR I:= 1 TO N DO
        BEGIN
                T1[I]:=(K0/V)*SQRT(SQR(X[I]-XD)+SQR(Y[I]-YD));
                FATIV[I]:=0;
        END;
        L:=0;
        FOR I:= 1 TO N-1 DO
        BEGIN
                FOR J:= I+1 TO N DO
                BEGIN
                        L:=L+1;
                        TAL[I,J]:=(K0/V)*SQRT(SQR(X[I]-X[J])+SQR(Y[I]-Y[J]));
                        D1:=K0*SQRT(SQR(X[I]-XD)+SQR(Y[I]-YD));
                        D3:=K0*SQRT(SQR(X[J]-XD)+SQR(Y[J]-YD));
                        D2:=K0*SQRT(SQR(X[I]-X[J])+SQR(Y[I]-Y[J]));
                        S[L]:=D1+D3-D2;
                        IA[L]:=I;
                        JA[L]:=J;
                        TAL[J,I]:=TAL[I,J];
                END;
        END;
```

```
END;

PROCEDURE RESULT;
BEGIN
        MI:=TRUNC((N-1)*N/2);
        FOR L:= 1 TO MI DO
                WRITELN ('L= ',L:2,' I= ',IA[L]:3,' J= ',JA[L]:3,' GANHO= ',S[L]:20,2);
END;

PROCEDURE ORDENA;
BEGIN
        STATUS:=FALSE;
        WHILE NOT STATUS DO
        BEGIN
                STATUS:=TRUE;
                MI:=TRUNC((N-1)*N/2.0);
                FOR L:= 1 TO MI-1 DO
                BEGIN
                        IF (S[L+1] > S[L]) THEN
                        BEGIN
                                STATUS:=FALSE;
                                I0:=IA[L+1];
                                J0:=JA[L+1];
                                S0:=S[L+1];
                                IA[L+1]:=IA[L];
                                JA[L+1]:=JA[L];
                                S[L+1]:=S[L];
                                IA[L]:=I0;
                                JA[L]:=J0;
                                S[L]:=S0;
                        END;
                END;
        END;
        (* RESULT;  *)
END;

PROCEDURE INICIA;
(* INICIALIZA OS VETORES REPRESENTATIVOS DO ROTEIRO *)
BEGIN
        NK:=0;NVA:=0;
        FOR I:= 1 TO N DO
        BEGIN
                VA[I]:=0;
                FOR K:= 1 TO N DO
                BEGIN
                        VT[I,K]:=0;VA[I]:=0;
                        NVT[K]:=0;DD[K]:=0.0;
                END;
        END;
END;

PROCEDURE NOVA_ROTA;
(* QUANDO OS NOS I E J NAO FOREM INCLUIDO EM NENHUM ROTEIRO,
UMA NOVO ROTEIRO SERA CRIADO. *)
BEGIN
```

```
        TT:=T1[XI]+T1[XJ]+TP*2+TAL[XI,XJ];
        WT:=W[XI]+W[XJ];
        IF (TT <= TC ) AND (WT <= WV) THEN
        BEGIN
                NK:=NK+1;
                NVT[NK]:=2;
                VT[NK,1]:=XI;
                VT[NK,2]:=XJ;
        END;
END;

PROCEDURE BUSCA_ROTA;
(* P/ DOIS NOS I E J VERIFICAR SE PERTENCEM AS ROTAS JA EXISTENTES
OU NAO E SE SAO PONTOS NAO INTERIORES A ESSAS ROTAS. *)
BEGIN
        FLAGI:=0;FLAGJ:=0;CSI:=-1;CSJ:=-1;
        IF (NK=0) THEN
        BEGIN
                CSI:=0;
                CSJ:=0;
        END ELSE
        BEGIN
                OK2:=FALSE;K:=0;CSI:=0;
                WHILE NOT OK2 DO
                BEGIN
                        K:=K+1;
                        NV:=NVT[K];
                        FOR M:= 1 TO NV DO
                        BEGIN
                                IF (XI = VT[K,M]) THEN
                                BEGIN
                                        CSI:=-1;
                                        IF (M=1) OR (M=NV) THEN CSI:=1;
                                        FLAGI:=K;
                                        IS:=M;
                                END;
                        END;
                        IF (CSI=1) OR (K=NK) THEN OK2:=TRUE;
                END;
                OK2:=FALSE;K:=0;CSJ:=0;
                WHILE NOT OK2 DO
                BEGIN
                        K:=K+1;
                        NV:=NVT[K];
                        FOR M:= 1 TO NV DO
                        BEGIN
                                IF (XJ=VT[K,M]) THEN
                                BEGIN
                                        CSJ:=-1;
                                        IF (M=1) OR (M=NV) THEN CSJ:=1;
                                        FLAGJ:=K;
                                        JS:=M
                                END;
                        END;
                        IF (CSJ=1) OR (K=NK) THEN OK2:=TRUE;
```

```
                END;
        END;
        IF (CSI=1) AND (CSJ=1) AND (FLAGI=FLAGJ) THEN
        BEGIN
                CSI:=-1;CSJ:=-1;
        END;
END;

PROCEDURE CICLO_A;
(* VERIFICA SE O TEMPO DE CICLO DE UM ROTEIRO POTENCIAL
RESPEITA AS RESTRICOES DE JORNADA DE TRABALHO E DE CAPACIDADE
DE CARGA. *)
BEGIN
        OK1:=FALSE;
        XI:=VA[1];XJ:=VA[NVA];
        TT:=T1[XI]+T1[XJ]+TP*NVA;
        WT:=W[XI];
        FOR M:= 2 TO NVA DO
        BEGIN
                XI:=VA[M-1];XJ:=VA[M];
                TT:=TT+TAL[XI,XJ];
                WT:=WT+W[XJ];
        END;
        IF (TT <= TC) AND (WT <= WV) THEN
        BEGIN
                OK1:=TRUE;NVT[K]:=NVA;
                FOR M:= 1 TO NVA DO
                BEGIN
                        VT[K,M]:=VA[M];
                        VA[M]:=0;
                END;
                NVA:=0;
        END;
END;

PROCEDURE CICLO_B;
(* CALCULA O TEMPO DE CICLO DOS ROTEIROS RESULTANTES, DISTANCIA
PERCORRIDA E CARGA TOTAL TRANSPORTADA PELO VEICULO. *)
BEGIN
        FOR K:= 1 TO NK DO
        BEGIN
                NV:=NVT[K];DD[K]:=0.0;
                IF (NV > 0) THEN
                BEGIN
                        NVA:=NV;
                        FOR M:= 1 TO NV DO
                        BEGIN
                                VA[M]:=VT[K,M];
                                IF (M = 1) THEN
                                DD[K]:=DD[K]+K0*SQRT(SQR(XD-
X[VA[M]])+SQR(YD-Y[VA[M]]))
                                ELSE    DD[K]:=DD[K]+K0*SQRT(SQR(X[VA[M]]-
X[VA[M-1]])+SQR(Y[VA[M]]-Y[VA[M-1]]))
                        END;
```

```
                    DD[K]:=DD[K]+K0*SQRT(SQR(XD-X[VA[NV]])+SQR(YD-
Y[VA[NV]]));
                    CICLO_A;
                    TX[K]:=TT;
                    WX[K]:=WT;
                END;
        END;
END;


PROCEDURE VERIFICA;
(* VERIFICA SE TODOS OS PONTOS FORAM INCLUIDOS EM ALGUM
ROTEIRO, CASO CONTRARIO, GERA ROTEIROS INDIVIDUALIZADOS. *)
BEGIN
        FOR M:=1 TO N DO
        VA[M]:=0;
        FOR K:= 1 TO NK DO
        BEGIN
                NV:=NVT[K];
                FOR M:= 1 TO NV DO
                BEGIN
                        MN:=VT[K,M];
                        VA[MN]:=1;
                END;
        END;
        FOR M:= 1 TO N DO
        BEGIN
                IF (VA[M]=0) THEN
                BEGIN
                        NK:=NK+1;NVT[NK]:=1;
                        VT[NK,1]:=M;
                        IF (T1[M]*2.0 + TP > TC) OR (W[M] > WV) THEN
FATIV[NK]:=-1;
                END;
        END;
END;

PROCEDURE C_WRIGHT;
(* METODO CLARKE-WRIGHT P/ OBTENCAO DOS ROTEIROS OTIMIZADOS. *)
BEGIN
        INICIA;
        PARAMETROS;
        ORDENA;
        FOR L:=1 TO MI DO
        BEGIN
                NVA:=0;
                FOR M:=1 TO N DO
                VA[M]:=0;
                XI:=IA[L];XJ:=JA[L];
                BUSCA_ROTA;
                IF (CSI=0) AND (CSJ=0) THEN NOVA_ROTA;
                (* CASO 1 *)
                IF (CSI=1) AND (CSJ=0) THEN
                BEGIN
                        K:=FLAGI;NVA:=NVT[K];
```

```
        IF (IS > 1) THEN
        BEGIN
                FOR M:= 1 TO NVA DO
                VA[M]:=VT[K,M];
                VA[M+1]:=XJ;
                NVA:=NVA+1;
                CICLO_A;
        END;
        IF (IS = 1) THEN
        BEGIN
                FOR M:= 1 TO NVA DO
                BEGIN
                        MN:=NVA+2-M;
                        VA[MN]:=VT[K,MN-1];
                END;
                VA[1]:=XJ;
                NVA:=NVA+1;
                CICLO_A;
        END;
END;
(* CASO 2 *)
IF (CSI=0) AND (CSJ=1) THEN
BEGIN
        K:=FLAGJ;NVA:=NVT[K];
        IF (JS > 1) THEN
        BEGIN
                FOR M:=1 TO NVA DO
                VA[M]:=VT[K,M];
                VA[M+1]:=XI;
                NVA:=NVA+1;
                CICLO_A;
        END;
        IF (JS = 1) THEN
        BEGIN
                FOR M:= 1 TO NVA DO
                BEGIN
                        MN:=NVA+2-M;
                        VA[MN]:=VT[K,MN-1];
                END;
                VA[1]:=XI;
                NVA:=NVA+1;
                CICLO_A;
        END;
END;
(* CASO 3 *)
IF (CSI=1) AND (CSJ=1) THEN
BEGIN
        K:=FLAGI;NVI:=NVT[K];
        IF (IS = 1) THEN
        BEGIN
                FOR M:=1 TO NVI DO
                BEGIN
                        MN:=NVI-M+1;
                        VA[M]:=VT[K,MN];
                END;
```

```
                        FOR M:=1 TO NVI DO
                        VT[K,M]:=VA[M];
                END;
                K1:=FLAGJ;NVJ:=NVT[K1];
                IF (JS > 1) THEN
                BEGIN
                        FOR M:=1 TO NVJ DO
                        BEGIN
                                MN:=NVJ-M+1;
                                VA[M]:=VT[K1,MN];
                        END;
                        FOR M:=1 TO NVJ DO
                        VT[K1,M]:=VA[M];
                END;
                FOR M:= 1 TO NVI DO
                VA[M]:=VT[K,M];
                FOR M:= NVI+1 TO NVI+NVJ DO
                VA[M]:=VT[K1,M-NVI];
                NVA:=NVI+NVJ;
                CICLO_A;
                IF OK1 THEN
                BEGIN
                        FOR M:= 1 TO N DO
                        VT[K1,M]:=0;
                        NVT[K1]:=0;
                END;
        END;
    END;
    VERIFICA
END;


PROCEDURE RESULTADOS;
(* APRESENTA, EM TELA, OS ROTEIROS OTIMIZADOS E RESPECTIVOS
TEMPOS DE CICLO E CARGA TRANSPORTADA. *)
BEGIN
    J:=0;
    FOR K:= 1 TO NK DO
    BEGIN
        NV:=NVT[K];
        IF (NV > 0) THEN
        BEGIN
            J:=J+1;
            TELA;
            GOTOXY(28,4);WRITE('* * * ROTEIRO NUMERO ',J:3,' * * *
');
            FOR M:= 1 TO NV DO
            BEGIN
                XI:=VT[K,M];
                GOTOXY(4*(M-1)+1,10);WRITE(XI:4);
            END;
            GOTOXY(1,13);WRITE('TEMPO DE CICLOS (HORAS):');
            GOTOXY(30,13);WRITE(TX[K]:10:1);
            GOTOXY(1,15);WRITE('CARGA TRANSPORTADA (TON):');
            GOTOXY(30,15);WRITE(WX[K]:10:1);
```

```
                        GOTOXY(1,17);WRITE('DISTANCIA TOTAL (KM):');
                        GOTOXY(30,17);WRITE(DD[K]:10:1);
                        IF (FATIV[K] < 0) THEN
                        BEGIN
                                GOTOXY(1,22);WRITE('ESTE ROTEIRO EXCEDE
LIMITE(S) - SOLUCAO IMPOSSIVEL!');
                        END;
                        GOTOXY(1,23);WRITE(LIN);
                        GOTOXY(25,24);WRITE('APERTE <ENTER> PARA
CONTINUAR');READLN(LC);
                END;
        END;
END;

PROCEDURE ROTEIRO;
(* ANALISA ROTEIROS SUBMETIDOS PELO USUARIO. *)
BEGIN
        STATUS:=TRUE;
        WHILE STATUS DO
        BEGIN
                TELA;
                GOTOXY(10,5);WRITE('DESEJA ANALISAR OUTROS ROTEIROS
(S/N)?');
                GOTOXY(60,5);READLN(LC);LC:=UPCASE(LC);
                IF (LC='N') THEN STATUS:=FALSE;
                IF STATUS THEN
                BEGIN
                        GOTOXY(10,7);WRITE('NUMERO DE PONTOS DE
PARADA DO ROTEIRO (EXCETO DEPOSITO):');
                        GOTOXY(70,7);READLN(NV);
                        GOTOXY(10,9);WRITE('DIGITE OS ',NV:3,' PONTOS DO
ROTEIRO:');
                        GOTOXY(1,11);
                        FOR M:= 1 TO NV DO
                        BEGIN
                                GOTOXY(3*(M-1)+3,13);READLN(VT[1,M]);
                        END;
                        NK:=1;NVT[NK]:=NV;CICLO_B;
                        RESULTADOS;
                END;
        END;
END;

BEGIN

        LIN:='';
        WHILE LENGTH(LIN) < 80 DO
        LIN:=LIN+LINHA;

        TELA;
        GOTOXY(10,4);WRITE('>>> OPCOES:');
        GOTOXY(10,7);WRITE('[A] DEMONSTRACAO COM DADOS EX 10.6');
        GOTOXY(10,8);WRITE('[B] ENTRADA DE DADOS PELO USUARIO');
        GOTOXY(10,12);WRITE('SELECIONE OPCAO:');
        GOTOXY(60,12);READLN(LC);LC:=UPCASE(LC);
```

```
        IF (LC='A') THEN DADOS ELSE
        BEGIN
                ENTER_1;
                ENTER_2;
        END;
        C_WRIGHT;
        CICLO_B;
        RESULTADOS;
        ROTEIRO;
END.
```