Sistema para Resolver o Problema de Roteamento e Inventário com Demanda Estocástica: Comparando Diferentes Heurísticas na Relaxação Lagrangeana

Alternative Title: System to Solve the Inventory Routing Problem with Stochastic Demand: Comparing Different Heuristics in the Lagrangian Relaxation

Pedro Yuri A. L. Alves
Universidade de São Paulo
pedro.yuri.alves@usp.br

Karina Valdivia Delgado
Universidade de São Paulo
kvd@usp.br

Alexandre da Silva Freire
Universidade de São Paulo
afreire@ime.usp.br

Valdinei Freire da Silva
Universidade de São Paulo
valdinei.freire@usp.br

RESUMO

Fornecedores necessitam atender a demanda de seus clientes da forma mais otimizada possível e mantendo a qualidade de seu serviço. Porém, em muitos casos essa demanda é desconhecida. O problema conhecido como *problema de roteirização e inventário com demanda estocástica* combina: (i) o controle de estoque; (ii) o transporte do produto; e (iii) decisões de agendamento da entrega considerando essa classe de demanda. Este trabalho tem como objetivo melhorar o algoritmo estado da arte baseado em programação matemática e relaxação lagrangeana visando encontrar soluções com custo menor. Para tal, foram propostas três variantes do algoritmo considerando diferentes heurísticas. Foram realizados experimentos com instâncias de teste contendo 15, 25 e 50 clientes; e foram analisados o custo final da solução e o tempo computacional para a solução convergir.

Palavras-Chave

Problema de Roteamento e Inventário, Roteamento de Veículos, Demanda Estocástica, Relaxação Lagrangeana.

ABSTRACT

Providers need to supply the demand of their clients as optimally as possible and maintaining the quality of their service. However, in many cases this demand is unknown. The problem known as inventory routing problem with stochastic demand combines: (i)inventory control; (ii) product transportation; and (iii) delivery scheduling decisions considering this type of demand. This work aims to improve the state of the art algorithm based on mathematical programming and lagrangian relaxation aiming to find solutions with lower cost. To accomplish this, three variants of the algorithm were proposed considering different heuristics. Experiments were performed with test instances containing 15, 25 and 50 clients; and the final cost of the solution and the computational time for the convergence were analyzed.

SBSI 2017 June 5^{th} – 8^{th}, 2017, Lavras, Minas Gerais, Brazil

CCS Concepts

•**Theory of computation** → **Integer programming;**

Keywords

Inventory Routing Problem, Vehicle Routing Problem, Stochastic Demand, Lagrangian Relaxation.

1. INTRODUÇÃO

Uma das prioridades dos fornecedores é satisfazer os clientes, buscando sempre meios de melhorar e, principalmente, reduzir seus custos da operação logística [3]. Os fornecedores preocupam-se com aspectos da cadeia de suprimentos que inclui o armazenamento, as rotas e as quantidades a serem entregues aos seus clientes. Problemas de otimização de cada um desses aspectos da cadeia de suprimentos foram apresentados ao longo dos anos, e sua evolução natural é a junção desses problemas para resolvê-los em conjunto.

Nos modelos mais tradicionais, os próprios clientes controlam seus estoques e quando é necessário reabastecer, o mesmo realiza o pedido ao fornecedor considerando suas necessidades naquele momento, restando ao fornecedor somente a responsabilidade de planejar as entregas. Esse problema é conhecido como o Problema de Roteamento de Veículos (em inglês Vehicle Routing Problem – VRP).

A união do gerenciamento de estoque, VRP e decisões do agendamento das entregas é conhecida como Problema de Roteamento e Inventários (em inglês Inventory Routing Problem – IRP) [8]. O IRP tem como objetivo otimizar os custos de estoque e custos que envolvem a quantidade, frequência de distribuição e as rotas adotadas para a realização da reposição de estoque. Uma boa solução para o IRP é de grande auxílio para diminuir o custo de logística de uma

empresa, pois promove eficiência na operação e melhoria na qualidade do serviço.

Os IRPs podem ser classificados em determinísticos ou estocásticos. Se a informação da demanda está totalmente disponível durante a tomada de decisão, isto é, no inicio do horizonte de planejamento, o problema IRP é determinístico. Caso somente a distribuição de probabilidade da demanda é conhecida, o problema é nomeado como Problema de Roteamento e Inventário Estocástico (em inglês Stochastic Inventory Routing Problem – SIRP). Os trabalhos propostos para solução do SIRP tem que lidar com a dificuldade de atender a estocasticidade e os futuros valores desconhecidos da demanda.

As soluções para SIRP foram categorizadas em [8] como soluções heurísticas, baseadas em processo de decisão Markoviano, e baseadas em programação matemática. A maioria dos trabalhos estão classificados como algoritmos heurísticos, por exemplo, [12] .Outros trabalhos estão baseados no processo de decisão Markoviano, como em [2]. Porém essa última abordagem é impraticável para instâncias de SIRP de tamanho médio e grande [12]. A última classe são dos trabalhos baseados em programação matemática, abordagem apropriada para trabalhar com incertezas, quando não há informação disponível sobre os parâmetros da distribuição de probabilidade da demanda [8].

Um dos trabalhos mais recentes na literatura que usa programação matemática é [1]. Esse trabalho utiliza relaxação lagrangeana para: (i) relaxar as restrições do problema original que são consideradas complexas para o modelo, (ii) estimar limitantes inferiores e superiores para a solução, e (iii) atualizar esses valores para obter uma solução viável e de menor custo. Essa abordagem demonstra um bom desempenho computacional, obtendo uma solução viável em poucos minutos. Para melhorar ainda mais a qualidade das soluções apresentadas, este trabalho tem por objetivo aplicar diferentes heurísticas na relaxação lagrangeana [1] para encontrar soluções com custo menor.

O restante deste texto está organizado da seguinte forma. A Seção 2 descreve de forma detalhada o algoritmo de programação matemática com relaxação lagrangeana para o SIRP proposto por Rahim [1]. A Seção 3 apresenta as variantes propostas para o algoritmo do Rahim. A Seção 4 apresenta o método utilizado neste trabalho e a Seção 5 apresenta uma discussão dos resultados obtidos. Por fim, a Seção 6 apresenta as conclusões.

2. ALGORITMO DE RAHIM

Em [1] é proposta a utilização da relaxação lagrangeana para determinar uma solução viável para o problema SIRP considerando múltiplos períodos. A Figura 1 apresenta todos os passos da modelagem e solução proposta em [1]. Primeiro é feita a modelagem estocástica, em que o SIRP é modelado como um problema de programação inteira mista e estocástico (PIME). Depois, é criado um novo modelo do problema, traduzindo as variáveis estocásticas para uma aproximação determinística. Posteriormente, o problema é decomposto em dois subproblemas, o subproblema de estoque e o subproblema de roteamento de veículos usando os multiplicadores lagrangeanos (passo descrito na Seção 2.3). O subproblema de estoque trata da parte estocástica da demanda e o subproblema de roteirização de veículos é resolvido através da programação inteira mista. E finalmente, um limite inferior e superior para o problema determinístico é encontrado usando o algoritmo subgradiente para atualizar os multiplicadores lagrangeanos. Esse último passo de cor rosa na Figura 1 é descrito na Seção 2.4.

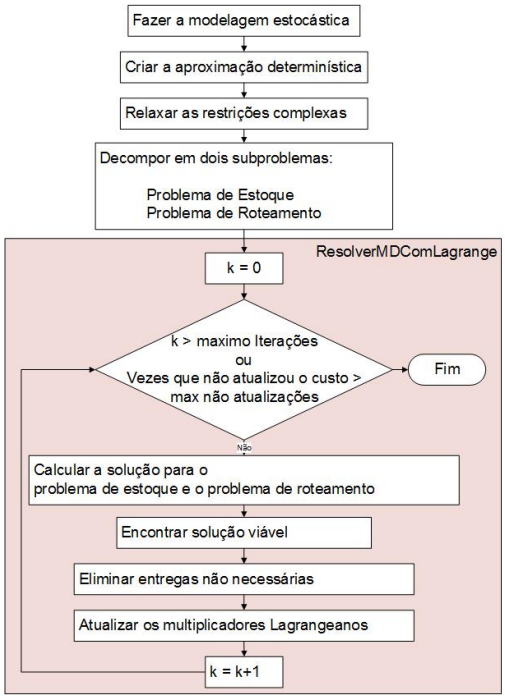


Figure 1: Passos da solução proposta em [1].

2.1 Modelagem Estocástica

As variáveis, parâmetros, restrições e função objetivo da PIME para modelar o SIRP, proposto por [1], são descritos a seguir. Seja $H = \{1, 2, \cdots, T\}$ o conjunto de períodos consecutivos e $H^+ = H \cup \{0\}$, o parâmetro τ_h representa o número de horas em um período h em que $h \in H^+$. Seja ζ o conjunto de clientes e $S = \zeta \cup \{r\}$, em que $r = 0$ é o depósito, e os clientes são numerados de 1 até N. A frota de veículos é homogênea e cada veículo $v \in V$ tem capacidade k_v. Para simplificar o entendimento do modelo proposto, vamos considerar que o conjunto S^2 é a combinação de todos os pares (i, j), em que $i, j \in S$. Os outros parâmetros do modelo são [1]:

- R_{rh}: quantidade de produto a ser reabastecida no estoque do depósito no período $h \in H$.
- d_{jh}: demanda estocástica do cliente $j \in \zeta$ por hora do período $h \in H$. Os autores assumem que essa demanda tem uma distribuição normal em que $D_j = E(d_{jh})$, e o desvio padrão é igual a σ_j.
- η_{jh} : custo de armazenamento de cada unidade do produto para $j \in S$ no período $h \in H$.
- ψ_v: custo operacional do veículo $v \in V$.
- φ_{jh}: custo fixo de entrega para $j \in S$ no período $h \in H$.
- δ_v: custo de viagem do veículo $v \in V$ por km.
- ν_v: velocidade média do veículo $v \in V$ (em km por hora).

- θ_{ij}: duração da viagem entre o par $(i,j) \in S^2$ (em horas).

Assim, a demanda estocástica do cliente $j \in \zeta$ no período $h \in H$ é $\Lambda_{jh} = d_{jh}\tau_h$. Considerando os últimos quatro parâmetros, o custo de transporte e de entrega de uma viagem entre o par $(i,j) \in S^2$ usando o veículo $v \in V$ no período $h \in H$ é $\Gamma_{ijhv} = \delta_v \nu_v \theta_{ij} + \varphi_{jh}$. As variáveis consideradas no modelo proposto por [1] são:

- x_{ijhv}: variável binária com valor 1 quando $j \in S$ é visitado imediatamente depois de $i \in S$ pelo veículo $v \in V$ no período $h \in H$, e 0 caso contrário.
- y_{hv}: variável binária com valor 1 quando o veículo $v \in V$ é utilizado no período $h \in H$, e 0 caso contrário.
- q_{jh}: quantidade do produto entregue para o cliente $j \in \zeta$ no período $h \in H$.
- I_{jh}: nível de estoque de $j \in S$ no final do período $h \in H^+$.
- Q_{ijhv}: quantidade do produto restante no veículo $v \in V$ quando visita $j \in S$ imediatamente após $i \in S$ no período $h \in H$. Quando não existe a viagem (i,j), o valor dessa variável é zero.

O modelo de PIME que resolve o SIRP é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \quad (1)$$

$s.a.:$

$$\sum_{v\in V}\sum_{i\in S} x_{ijhv} \le 1, \quad \forall j \in \zeta, h \in H, \quad (2)$$

$$\sum_{i\in S}(x_{ijhv} - x_{jihv}) = 0, \quad \forall j \in S, h \in H, v \in V, \quad (3)$$

$$\sum_{(i,j)\in S^2}\theta_{ij}x_{ijhv} \le \tau_h, \quad \forall h \in H, v \in V, \quad (4)$$

$$\sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv}) = q_{jh}, \quad \forall j \in \zeta, h \in H, \quad (5)$$

$$Q_{jkhv} \le k_v x_{jkhv}, \quad \forall i,j \in S, h \in H, v \in V, \quad (6)$$

$$I_{r,h-1} + R_{rh} - I_{r,h} = \sum_{j\in\zeta} q_{jh}, \quad \forall j \in \zeta, h \in H, \quad (7)$$

$$I_{j,h-1} + q_{jh} - I_{j,h} = \Lambda_{jh}, \quad \forall j \in \zeta, h \in H, \quad (8)$$

$$I_{j0} \le I_{jT}, \quad \forall j \in \zeta, \quad (9)$$

$$x_{rjhv} \le y_{hv} \quad \forall j \in \zeta, h \in H, v \in V, \quad (10)$$

$$x, y \in \{0,1\}^{|S|^2|H||V|}, I \ge 0, Q \ge 0, q \ge 0, \quad (11)$$

em que a função objetivo é composta de três custos: (i) o custo de operação do veículo; (ii) o custo total de transporte e de entrega; e (iii) o custo de armazenamento do produto no final de cada período.

A Restrição (2) garante que no período h cada cliente seja visitado no máximo uma vez. A Restrição (3) assegura que a rota de um veículo deve começar no depósito, passar pelos clientes e retornar para o depósito, assegurando o ciclo da rota. A Restrição (4) garante que todo o tempo de trajeto da rota do veículo não ultrapasse as horas de trabalho do período. A Restrição (5) determina a quantidade de entrega para um cliente. A Restrição (6) assegura que a capacidade do veículo não vai ser ultrapassada, e que as variáveis Q_{ijhv} não possam transportar qualquer quantidade de produto a menos que x_{ijhv} tenha valor 1. A Restrição (7) assegura o balanceamento do estoque do depósito e a Restrição (8) atende as demandas estocásticas dos clientes. Uma vez que não há custo para o inventário inicial, a Restrição (9) garante que o estoque de um determinado cliente no final do horizonte de planejamento não seja menor do que o do início do planejamento. E por fim, a Restrição (10) assegura que um veículo não pode atender um cliente, em um determinado período h, se o mesmo não for selecionado no período h. Essa é a restrição que relaciona a utilização do veículo com o atendimento dos clientes.

2.2 Aproximação Determinística

No modelo determinístico proposto por [1] que aproxima a PIME (Modelo (1)), a Restrição (8) é substituída pela Restrição (12) considerando um nível de confiança de $(1-\alpha)$ de satisfazer a demanda estocástica em cada período.

$$I_{j,h-1} + \sum_{s=h}^{T} q_{js} \ge \sum_{s=h}^{T} E(d_{js})\tau_s + z_\alpha\left(\sqrt{T-h+1}\right)\sigma_j, \quad (12)$$

$\forall j \in \zeta, h \in H.$

A Restrição (12) previne a escassez de estoque em cada cliente com um nível de confiança $(1-\alpha)$. Assim, $100*(1-\alpha)\%$ do nível de serviço é garantido, definido pelo valor normal padrão z_α. O modelo determinístico resultante (MD) é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} \quad (13)$$

$s.a: (2)-(7), (9), (10)-(12)$

2.3 Usando Relaxação Lagrangeana

Em [1] é proposto um algoritmo que usa relaxação lagrangeana para decompor o problema e encontrar limitantes inferiores (LI) e superiores (LS) para o problema. A relaxação Lagrangeana consiste em relaxar algumas restrições de um modelo computacionalmente difícil de resolver, e dessa forma o modelo computacionalmente complexo pode ser convertido para um modelo tratável. Essas restrições são eliminadas e incorporadas na função objetivo como uma penalidade. A restrição (5) foi escolhida por [1] para ser relaxada pois combina variáveis de estoque (q_{jh}) e variáveis de transporte (Q_{ijhv}). Assim, a restrição (5) é inserida na função objetivo com multiplicadores lagrangeanos μ_{jh} para todo $j \in S, h \in H$. Essa reformulação que usa relaxação lagrangeana, chamada de RL, é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left(q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right) \quad (14)$$

$s.a: (2)-(4), (6), (7), (9), (10)-(12).$

No Modelo (14) podemos formar dois conjuntos disjuntos de restrições considerando o tipo de variáveis usadas. As Restrições (7),(9) e (12) contém apenas variáveis de inventário e as restrições (2)-(4),(6) e (10) contém apenas variáveis de roteamento. Se observarmos a função objetivo, é possível também separar os termos em dois conjuntos, um que contem os temos das variáveis de inventário e outro com as variáveis de roteamento. Assim, o problema de IRP pode ser decomposto em um subproblema de estoque (I) e um subproblema de roteamento (R). Apesar de que o problema de roteamento e o problema de inventário resultantes são NP-difíceis [5],eles podem ser resolvidos na prática mais facilmente por um otimizador de programação inteira mista. Esses dois submodelos são apresentados a seguir.

O modelo do problema de inventário (RLI) é:

$$min\ Z_{RLI} = \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}q_{jh} \quad (15)$$
$$s.a: (7), (9)\, e\, (12).$$

O modelo do problema de roteamento (RLR) é:

$$min\ Z_{RLR} = \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) - \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[\sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right] \quad (16)$$
$$s.a: (2) - (4), (6)\, e\, (10)$$
$$E(d_{j1})\tau_1 + z_\alpha\sigma_j - I_{j0} \le \sum_{v\in V}\sum_{i\in S}Q_{ij1v}, \quad \forall j\in\zeta. \quad (17)$$

A Restrição (17), garante com certo nível de confiança que não ocorrerá escassez em cada cliente durante o primeiro período no horizonte de planejamento.

Partindo da decomposição descrita anteriormente, um limitante inferior (LI) para o Modelo (13) pode ser encontrado para qualquer vetor de multiplicadores lagrangeano μ. O melhor LI pode ser calculado a partir do vetor de multiplicadores lagrangeano ótimo que é a solução do dual do Modelo (14), i.e.:

$$max \quad L(\mu_{jt}), \quad (18)$$

em que $L(\mu_{jt})$ é:

$$min \sum_{h\in H}\sum_{v\in V}\left(\psi_v y_{hv} + \sum_{(i,j)\in S^2}\Gamma_{ijhv}x_{ijhv}\right) + \sum_{h\in H^+}\sum_{j\in S}\eta_{jh}I_{jh} + \sum_{h\in H}\sum_{j\in\zeta}\mu_{jh}\left[q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv})\right] \quad (19)$$

2.4 Resolvendo o Problema Dual

Para resolver o Modelo (18), primeiramente os subproblemas de inventário (Modelo (15)) e de roteamento (Modelo (16)) são solucionados para encontrar um limitante inferior, e então o algoritmo de subgradiente é utilizado para ajustar o valor de μ_{jt} ao longo do tempo para encontrar valores que produzem melhores limitantes inferiores. Além disso, uma heurística que recebe as soluções da relaxação lagrangeana é usada para compor uma solução viável do problema original.

2.4.1 Algoritmo Subgradiente

O processo de otimização do subgradiente (Algoritmo 1) que ajusta μ_{jt}, gera limitantes inferiores e superiores a cada iteração do algoritmo, atualizando o melhor limite inferior e superior até o momento. Dado um valor inicial μ^0, uma sequencia de multiplicadores lagrangeanos é gerada usando a equação:

$$\mu^{k+1} = \mu^k + s^k g^k, \quad (20)$$

em que g^k é a matriz subgradiente e s^k é o tamanho do passo para atualização de μ. Essa matriz é calculada usando a seguinte equação:

$$g^k_{jh} = q_{jh} - \sum_{v\in V}\sum_{i\in S}(Q_{ijhv} - Q_{jihv}), \quad (21)$$

em que $j\in\zeta$ e $h\in H$; e q_{jh}, Q_{ijhv} e Q_{jihv} são os valores na solução encontrada para o problema na iteração k. O tamanho do passo é determinado considerando a seguinte equação:

$$s^k \leftarrow \omega(LS - Ck_{LI})/(\parallel g^k \parallel)^2, \quad (22)$$

em que ω é a *agilidade* do subgradiente, LS é um limitante superior para a solução ótima do problema original que pode ser encontrada aplicando uma heurística e Ck_{LI} é o custo usando os multiplicadores lagrangeanos na iteração k. Se $s^k \to 0$ e $\sum_{i=0}^{k} s^k \to \infty$ então Ck_{LI} converge para a solução do Modelo (18). Garantir essas condições pode ser uma tarefa difícil. Uma opção é inicializar ω com 2 e dividir esse valor por 2 quando o valor CK_{LI} não for melhorado. Essa opção apresenta bons resultados na prática, porém não garante as condições para a convergência [10].

O Algoritmo 1 recebe como parâmetros o número máximo de iterações, o número máximo de não atualizações do custo que pode ocorrer e os parâmetros do problema $\mathcal{P} = \langle H^+, \tau, S, V, k, R, d, \eta, \psi, \varphi, \delta,\ \nu, \theta, \Lambda, \Gamma\rangle$. A saída do algoritmo é a melhor solução encontrada $\mathcal{R} = \langle x, y, I, Q, q\rangle$. O algoritmo começa inicializando as variáveis LI, LS, ω e μ, essas duas últimas variáveis com valores iniciais entre zero e um (Linhas 3–6). Após a inicialização, $\mathcal{R}^*$ (a melhor solução viável encontrada até o momento) é calculada resolvendo o modelo MD (Modelo (13)) para um cenário em que cada cliente é atendido separadamente por um veículo (Linhas 8–9). O primeiro valor para LS é o custo obtido usando essa solução (Linha 10).

Em seguida, o algoritmo calcula LI, LS e atualiza os multiplicadores lagrangeanos μ em cada iteração (Linhas 11–38). O algoritmo só irá ser finalizado quando o número máximo de iterações for alcançado (Linha 11) ou o custo Ck_{LI} não for melhorado certo número de vezes (Linhas 35–37).

Para ajustar o LI são resolvidos o subproblema de inventário (Modelo 15) e o subproblema de roteamento (Modelo 16) (Linhas 13 e 14). Se a somatória das soluções para esses dois subproblemas for maior que o LI atual, então LI é atualizado com esse novo valor, caso contrário ω é atualizado (Linhas 16–21).

Para ajustar o LS, uma heurística é aplicada para encontrar uma solução viável baseando-se no resultado obtido na solução dos dois subproblemas (Linha 23). Caso essa solução melhore o LS atual, então a solução encontrada é considerada a melhor solução até o momento (Linhas 25–28). Após isso, um procedimento de ajuste é aplicado para eliminar entregas não necessárias e LS é atualizado (Linhas 29–30). E por fim, os multiplicadores lagrangeanos μ são atualizados (Linhas 32–34).

2.4.2 Encontrar uma Solução Viável

O algoritmo EncontrarSoluçãoViável recebe a solução encontrada do modelo RLI e os parâmetros $\mathcal{P}$. A informação de quantidade que deve ser entregue em cada cliente durante cada período é utilizada para resolver o problema de roteamento. Esse problema é resolvido para cada período separadamente, utilizando um método heurístico de economias conhecido com Clarke & Wright (C&W) apresentado em [7]. O algoritmo também verifica que a solução encontrada não viole as restrições (4) e (6).

O C&W [7] é um dos algoritmos heurísticos mais conhecidos para VRP. O algoritmo monta uma lista de economias de custo obtidas pela união de duas rotas. Inicialmente, o pior cenário possível é considerado, no qual os clientes i e j são visitados em rotas separadas. Em seguida o método faz a combinação dois a dois de todos os pontos (clientes e centro de distribuição) e calcula a lista de economias que contem os ganhos de todos os pares. É feita a ordenação dessa lista em ordem decrescente, e a partir dessa ordena-

Algoritmo 1: ResolverMDComLagrange [1]

Entrada: maxI: máximo número de iteracoes, maxNA: máximo número de não atualizacões, $\mathcal{P}$: parâmetros do problema
Saída: $\mathcal{R}^*$: Melhor solução encontrada

```
início
    //**********Inicializando Variáveis
    LI ← 0
    LS ← ∞
    ω ← obterAleatório(0, 1)
    μ^k ← obterMatrizAleatória(0, 1)
    //**********Inicializando LS
    PSimples ← criarCenarioUmClientePorVeiculo(P)
    R* ← solução do modelo MD para PSimples
    LS ← custo da solução R* de MD
    para k = 1 to maxI faça
        //**********Atualizando LI
        R_R ← solução do modelo RLR com μ^k para P
        R_I ← solução do modelo RLI com μ^k para P
        Ck_LI ← custo da solução R_R de RLR +
          custo da solução R_I de RLI
        se Ck_LI > LI então
            LI ← Ck_LI
        fim
        else
            ω ← ω/2
        end
        //**********Atualizando LS
        R^k ← encontrarSoluçaoViável(R_I,P)
        Ck_LS ← custo da solução R^k de MD
        se Ck_LS > LS então
            LS ← Ck_LS
            R* ← R^k
        fim
        R* ← eliminarEntregasNãoNecessárias(R*,P)
        LS ← custo da solução R* de MD
        //**********Atualizando μ
        g^k ← calcularSubgradiente(R*)
        s^k ← ω(LS − Ck_LI)/(‖ g^k ‖)^2
        μ^k ← μ^k + s^k g^k
        se vezes não atualizou Ck_LI > maxNA então
            break
        fim
    fim
fim
retorna R*
```

ção os pontos são analisados. A união de pares é realizada considerando as restrições do problema e caso o ganho seja maior que zero, a união é considerada e colocada na lista de economias.

2.4.3 Eliminar Entregas não Necessárias

O Algoritmo eliminarEntregasNãoNecessárias tenta combinar duas ou mais entregas em todo o horizonte de planejamento, de um determinado cliente, em uma única entrega. Se essa combinação resulta em uma solução melhor do que a atual, isto é com menor custo, então ela é considerada a melhor solução até o momento.

3. VARIANTES PROPOSTAS PARA O ALGORITMO DE RAHIM

O algoritmo de Rahim, descrito na Seção 2, propõe uma solução viável para o SIRP utilizando a relaxação lagrangeana. Nesse algoritmo o método EncontrarSoluçãoViável é utilizado para atualizar o LS da solução. Conforme descrito na Subseção 2.4.2, em cada iteração, o algoritmo resolve o problema de roteamento de veículo para cada período utilizando um método heurístico de economias que também verifica que as Restrições (4) e (6) não sejam violadas.

Com o objetivo de melhorar a solução encontrada, este trabalho apresenta três variações do algoritmo do Rahim, chamadas de Rahim com Centroide (RCt), Rahim com *Simulated Annealing* (RSA) e Rahim com *Monte Carlo Savings* (RMC). Nessas variações, é trocado o método heurístico baseado em economias por outros métodos conhecidos na literatura a saber: o algoritmo *baseado em Centroide* [15], o algoritmo *Simulated Annealing* [11] e o algoritmo *Monte Carlo Savings* [9], respectivamente. Cada um desses algoritmos foi implementado e adaptado para verificar que as Restrições (4) e (6) não sejam violadas. Esses algoritmos são descritos a seguir.

3.1 Algoritmo Baseado em Centroide

Em [15] é proposto um algoritmo heurístico baseado em centroide para resolver o VRP. O algoritmo consiste em três fases: construção de clusters, ajuste de clusters e estabelecimento de rota. Na primeira fase, são construídos grupos (clusters) de clientes. O grupo é construído selecionando o cliente mais distante do depósito como semente do grupo. Após isso, são selecionados os clientes mais próximos do centroide do cluster, sempre atualizando a localização do centroide ao adicionar um novo cliente. Esse processo continua até a capacidade do veículo não ser ultrapassada. Quando isso acontece, a construção do grupo é finalizada, uma nova semente é selecionada e o processo de construção de um novo grupo começa novamente até que todos os clientes estejam em um grupo.

A segunda fase, ajusta os grupos formados com auxílio do centro geométrico dos grupos. Essa etapa verifica se um cliente está mais próximo do centroide de um grupo vizinho do que o centroide de seu grupo atual. Caso isso aconteça e a capacidade do veículo do grupo vizinho não seja violada, o cliente é movido para o grupo vizinho e os centroides são recalculados. Na última fase, as rotas são estabelecidas para o atendimento dos clientes de cada grupo. Para isso a heurística apresentada em [14] é usada para encontrar o caminho mais curto.

3.2 Algoritmo Simulated Annealing

Em [11], o método heurístico *Simulated Annealing* foi adaptado para o problema de roteamento de veículo. Cada possível combinação de rotas é chamada de configuração e o algoritmo começa por selecionar uma configuração inicial e em cada iteração do algoritmo é gerada uma nova configuração vizinha. Essa geração é desenvolvida usando dois processos de transformações a fim de explorar as soluções vizinhas.

A primeira transformação, chamada de *mover*, implica em encontrar cinco pares de clientes que tenham as distâncias mais próximas entre si, incluindo o depósito. Em seguida são selecionados outros cinco clientes aleatórios. Estes clientes são removidos de suas rotas e inseridos em rotas aleatórias se a demanda de entrega não ultrapassar a capacidade do veículo.

A segunda transformação, nomeada de *substituir a média mais alta*, calcula a distância média de cada par de clientes, seleciona os cinco clientes com a maior distância média e remove esses de suas rotas. A transformação seleciona as próximas cinco rotas aleatórias e insere os cinco clientes selecionados na rota que resulte em menor custo de atendimento.

A solução é sempre viável pois as transformações usam

uma abordagem construtiva que gera rotas viáveis. Se o custo da configuração vizinha (explorada na iteração) for melhor do que a da melhor solução encontrada até o momento, a configuração vizinha se torna a nova melhor solução. O algoritmo continua até que o tempo máximo de processamento seja alcançado ou até que o parâmetro de temperatura usado no algoritmo seja muito baixa.

3.3 Algoritmo Monte Carlo Savings

O algoritmo Monte Carlo Savings é baseado no algoritmo Clarke & Wright e usa técnicas de Simulação de Monte Carlo. Diferente dos outros trabalhos que unem essas duas técnicas, o estudo apresentado em [9] utiliza as Simulações de Monte Carlo para variar a ordem em que as arestas da lista de economias são percorridas. Dessa forma são alteradas as soluções finais e é permitido que diferentes rotas sejam geradas em cada simulação.

O algoritmo executa $r = 2000$ simulações, sendo que em cada uma delas é gerada uma lista de economias. Para cada lista gerada, o algoritmo procede como o método de Clarke & Wright. A lista é percorrida de forma decrescente e fundindo as rotas viáveis. Para permitir a variação nas listas de economias, a fórmula de cálculo de economia recebe um componente aleatório, o qual é usado para bonificar ou penalizar a economia gerada para a aresta.

4. GERADOR DE INSTANCIAS

Não há um benchmark padrão de instâncias de testes para ser utilizado nos experimentos para o problema SIRP [2, 13]. Alguns autores criaram suas próprias instâncias entre eles [4, 17, 18], outros autores acabaram utilizando instâncias de outros problemas como VRP e IRP já existentes na literatura, e as adaptaram para serem estocásticas como [6, 16]. Por esse motivo, as comparações entre técnicas desenvolvidas por diferentes autores, acaba se tornando inviável e dificilmente praticada nesse ramo da literatura.

Neste trabalho, é criada uma ferramenta de geração de instâncias padronizada conforme a geração proposta por [17]. Esse gerador de instâncias recebe um arquivo com os principais parâmetros do cenário, que são: (i) nome de identificação da instância; (ii) $posX$, vetor de posições no eixo x dos clientes; (iii) $posY$, vetor de posições no eixo y dos clientes; (iv) $d_{jh}Min$, valor mínimo possível da demanda $\forall j \in \zeta, h \in H$; (v) $d_{jh}Max$, valor máximo possível da demanda $\forall j \in \zeta, h \in H$; (vi) C, vetor com o nível máximo de estoque de cada cliente; (vii) N, T e |V|; I_{j0}, $\forall j \in \zeta$; $\tau_h, \forall h \in H$; v_v, k_v e δ_v, $\forall v \in V$; η_{jh} e φ_{jh}, $\forall j \in \zeta, h \in H$.

Note que, o arquivo de parâmetros de cenário contém valores mínimos e máximos possíveis para a demanda de cada cliente em cada período. Baseados nesses valores, o gerador de instâncias gera de maneira aleatória um valor para cada período no horizonte de planejamento, para depois calcular $D_j = E(d_{jh})$ e o desvio padrão σ_j. É realizado também o cálculo de outros atributos como a distância e tempo do trajeto entre cada par de clientes. Além disso, diferente do IRP determinístico não é possível determinar uma solução exata para uma instância devido à estocasticidade do SIRP. Porém, é possível determinar um limite inferior e um limite superior da solução, informação que pode ser usada para comparação entre técnicas distintas e que pode ser colocada no arquivo da instância.

Finalmente, um arquivo de instância em um formato padrão de texto é gerado. Esse arquivo contém o posicionamento dos clientes e do centro de distribuição; distancias e tempos de trajeto; a média e desvio padrão da demanda por cliente; e LI e LS do custo da solução.

5. RESULTADOS

5.1 Instâncias

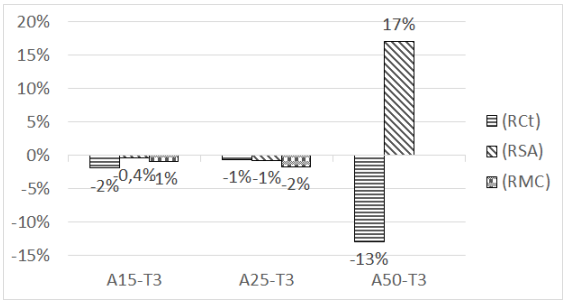
Foram considerados três conjuntos de instâncias diferentes ($N = 15, T = 3$), ($N = 25, T = 3$) e ($N = 50, T = 3$); em que N é o número de clientes e T o tamanho do horizonte de planejamento. Esses conjuntos foram criados pelo nosso gerador de instâncias utilizando os mesmos parâmetros que [1] e cada instância foi nomeada no formato (Ax-y-Tz), em que "x" é o total de clientes na instância, "y" é identificador de sequencia, e "z" representa o tamanho do horizonte de planejamento. O conjunto com N=15 foi criado considerando que a localização dos clientes é gerada randomicamente em um quadrado de 30 por 30 quilômetros e o centro de distribuição está localizado no meio do quadrado. A demanda d_{jh} é gerada randomicamente entre 1 e 3 toneladas por hora. O número de horas por período é $\tau_h = 8$ e o valor de z_α é $1,64$. O custo de armazenamento η_{jh} é gerado randomicamente entre $0,1$ e $0,15$ reais por tonelada, e o custo de entrega ϕ_{jh} com valor de 25 reais é o mesmo para cada cliente. O custo operacional ψ_v dos veículos é de 50 reais, e o custo de viagem δ_v é de 1 real por km/h. A frota tem veículos homogêneos com capacidade de 60 toneladas com uma velocidade média de 50 km/h.

Alguns parâmetros foram modificados para a criação das instâncias com N=25 e N=50. Os clientes são randomicamente posicionados em um quadrado de 100 por 100 e 200 por 200 quilômetros para N=25 e N=50, respectivamente. O centro de distribuição se mantém na posição central do quadrado e a demanda d_{jh} é gerada randomicamente entre 0.1 e 3 toneladas por hora. O custo de entrega ϕ_{jh} é gerado com valor de 10 reais para cada cliente, e o custo operacional ψ_v dos veículos é de 30 reais. Os outros parâmetros mantém o mesmo valor dos conjuntos de 15 clientes.

5.2 Discussão

As Tabelas 1, 2 e 3 mostram os resultados obtidos para ($N = 15, T = 3$), ($N = 25, T = 3$) e ($N = 50, T = 3$), respectivamente. Para cada instância das tabelas são apresentados o LI e LS para o algoritmo, o tempo computacional da solução em segundos ($CPU(s)$) e o $Gap = \frac{LS-LI}{LS} \times 100\%$. Nas tabelas, aqueles algoritmos que não terminaram de executar em até 5 horas estão marcados como *Não Terminou* (NT). Nesta seção, o algoritmo original de Rahim, o qual usa o algoritmo C&W, será chamado de RCW. Na Figura 2 é mostrada a porcentagem de melhoria no limitante superior e no tempo dos algoritmos propostos com relação a RCW.

O algoritmo do Rahim com centroide obteve resultados insatisfatórios considerando o LS em comparação com os demais algoritmos. Na maioria dos testes o custo final obtido ficou maior que o esperado quando comparado com RCW, em média RCt no LS foi 2%, 1% e 13% pior para as instâncias N=15, N=25 e N=50, respectivamente. Um ponto positivo para esse método é que ele obteve o menor tempo computacional médio nas instâncias de N=15 e N=50, sendo mais rápido em 42% e 6% respectivamente do que RCW. Já para N=25 essa abordagem foi 34% mais devagar. Note que, esse método depende da dispersão da localização dos clientes. Assim, é provável que em instâncias em que é possível observar uma clara separação de grupos de clientes, o algoritmo RCt obtenha melhores resultados, o que parece ter



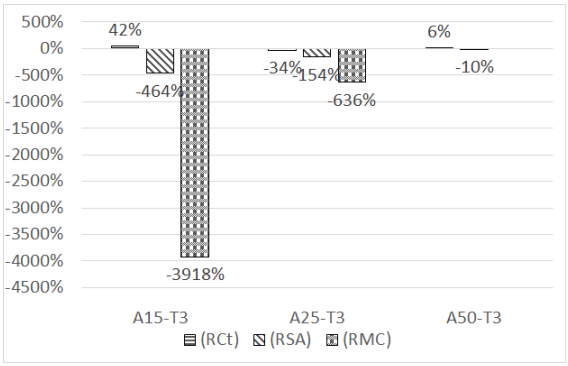


Figure 2: Porcentagem de melhoria no limite superior e no tempo com relação ao algoritmo RCW

Table 1: Resultados das instâncias (N = 15, T = 3).

Instância	Limitante Inferior				Limitante Superior				Gap (%)				Tempo de CPU (s)			
	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)
A15-0-T-3	**904**	902	**904**	**904**	**1080**	1122	**1080**	1122	**16,28**	19,64	**16,28**	19,43	**41**	59	249	4636
A15-1-T-3	**1387**	1378	1381	**1387**	**1749**	1760	**1749**	**1749**	20,73	21,73	21,07	**20,71**	121	**26**	274	5647
A15-2-T-3	1395	**1397**	1395	1395	**1663**	1700	**1663**	**1663**	**16,14**	17,83	**16,14**	**16,14**	**156**	167	1136	9775
A15-3-T-3	1442	**1443**	1442	1442	**1839**	1876	**1839**	**1839**	**21,58**	23,07	**21,58**	**21,58**	**104**	126	747	5888
A15-4-T-3	**1304**	**1304**	1302	**1304**	**1656**	1751	1706	1731	**21,28**	25,52	23,71	24,66	**33**	39	251	1787
A15-5-T-3	1337	**1338**	1337	1335	**1685**	1722	**1685**	**1685**	**20,69**	22,31	**20,69**	20,78	323	**27**	1669	2268
A15-6-T-3	1412	1401	1412	**1413**	**1779**	1798	**1779**	1784	**20,66**	22,10	**20,66**	20,83	191	**22**	812	7586
A15-7-T-3	1358	1357	**1359**	1358	1668	**1652**	1668	1668	18,57	**17,86**	18,52	18,57	**33**	35	300	1677
A15-8-T-3	**1379**	1362	**1379**	**1379**	1739	**1715**	1739	1739	20,69	**20,60**	20,69	20,69	32	**21**	247	1894
A15-9-T-3	954	**957**	955	956	**1001**	1062	1009	1015	**4,73**	9,86	5,33	5,74	**57**	109	471	2674
Média	**1287**	1284	1286	**1287**	**1586**	1616	1592	1600	**18,13**	20,05	18,47	18,91	109	**63**	616	4383

Table 2: Resultados das instâncias (N = 25, T = 3).

Instância	Limitante Inferior				Limitante Superior				Gap(%)				Tempo de CPU (s)			
	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)
A25-0-T-3	873	844	**877**	869	1162	1206	1162	**1144**	24,93	30,01	24,56	**23,99**	**525**	1061	1249	3329
A25-1-T-3	1132	**1134**	1125	1129	2013	**1633**	2090	2072	43,76	**30,56**	46,16	45,51	175	**164**	316	2794
A25-2-T-3	987	988	**991**	988	**1360**	1543	1388	1431	**27,43**	35,94	28,61	30,97	**194**	210	413	847
A25-3-T-3	**1140**	1139	**1140**	**1140**	1741	**1671**	1688	1688	34,50	**31,81**	32,48	32,48	269	**155**	492	1615
A25-4-T-3	**993**	**993**	**993**	992	1292	**1133**	1272	1238	23,14	**12,36**	21,92	19,88	**437**	938	897	1366
A25-5-T-3	1060	1061	**1064**	1058	1683	1793	**1491**	1829	37,05	40,80	**28,64**	42,15	**153**	231	474	4606
A25-6-T-3	999	999	1000	**1002**	**1140**	1422	1422	1219	**12,32**	29,72	29,69	17,79	363	**341**	784	1409
A25-7-T-3	1000	998	**1002**	1001	1582	1707	**1492**	1626	36,76	41,52	**32,83**	38,47	208	**198**	612	2017
A25-8-T-3	841	839	**842**	841	1347	**1171**	1425	1225	37,56	**28,36**	40,90	31,38	**386**	453	964	2487
A25-9-T-3	**945**	**945**	944	**945**	1395	1525	**1391**	1489	32,25	38,03	**32,12**	36,50	231	**201**	1273	1186
Média	997	994	**998**	996	**1472**	1481	1482	1496	**30,97**	31,91	31,79	31,91	**294**	395	747	2166

Table 3: Resultados das instâncias (N = 50, T = 3).

Instância	Limitante Inferior				Limitante Superior				Gap (%)				Tempo de CPU (s)			
	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)	(RCW)	(RCt)	(RSA)	(RMC)
A50-0-T-3	3497	3497	**3505**	NT	4882	5254	**4445**	NT	39,61	50,24	**26,82**	NT	**3677**	5656	5531	NT
A50-1-T-3	3534	3541	**3567**	NT	5050	5875	**4333**	NT	42,90	65,91	**21,47**	NT	5050	4870	**3774**	NT
A50-2-T-3	**3596**	3584	3568	NT	5540	6101	**4807**	NT	54,06	70,23	**34,73**	NT	**2852**	4321	6600	NT
A50-3-T-3	3345	**3366**	3357	NT	5276	5918	**4469**	NT	57,73	75,82	**33,12**	NT	2777	**2565**	2803	NT
A50-4-T-3	**3519**	**3519**	3516	NT	5409	5977	**4605**	NT	53,71	69,85	**30,97**	NT	4954	**4619**	7877	NT
A50-5-T-3	3555	3561	**3563**	NT	5800	6198	**4842**	NT	63,15	74,05	**35,90**	NT	3530	**1112**	3474	NT
A50-6-T-3	3528	**3552**	3520	NT	5228	6341	**4385**	NT	48,19	78,52	**24,57**	NT	6435	**3668**	5354	NT
A50-7-T-3	**4024**	4011	4015	NT	6098	6166	**4790**	NT	51,54	53,73	**19,30**	NT	6247	**6006**	6020	NT
A50-8-T-3	3456	**3480**	3446	NT	4925	6361	**4382**	NT	42,51	82,79	**27,16**	NT	**4193**	4296	5279	NT
A50-9-T-3	3725	3721	**3732**	NT	5088	5998	**4468**	NT	36,59	61,19	**19,72**	NT	3255	3252	**1426**	NT
Média	3578	**3583**	3579	NT	5330	6019	**4553**	NT	49,00	68,23	**27,38**	NT	4297	**4037**	4814	NT

acontecido nas instâncias A15-7-T-3 e A15-8-T-3.

O algoritmo Rahim com Simulated Annealing apresentou os melhores resultados quando comparado ao algoritmo RCW. Para N=15 e N=25 o algoritmo *RSA* encontra soluções com custo em média 0,4% e 1% maior que o algoritmo *RCW*. Porém, são nas instâncias com N=50 que essa abordagem se destaca, obtendo os menores custos entre todos os algoritmos, em média 17% menores. O algoritmo *RSA* é 464% mais devagar para N=15, 154% para N=25 e de apenas 10% para N=50.

Por último, os resultados de LS e LI para N=15 e N=25 usando o algoritmo do Rahim com Monte Carlo Savings fo-

ram muito parecidos com os do algoritmo de *RCW*. Apenas algumas mínimas diferenças foram notadas nos resultados, sendo de 1% para N=15 e de 2% para N=25. O tempo de processamento computacional é a desvantagem desse método, entre todas as abordagens implementadas, o algoritmo *RMC* apresentou o maior tempo computacional para terminar os experimentos para N=15 e N=25; e para as instâncias de N=50 este algoritmo não terminou em menos de 5 horas.

6. CONCLUSÃO

Neste trabalho foram propostas três variações para o algoritmo de Rahim modificando o método EncontrarSoluçãoViável. Esse método, que busca soluções viáveis para o problema de roteamento de veículo em cada período do planejamento, foi substituído por uma adaptação de três algoritmos conhecidos de roteamento de veículos que inclui a verificação para que as Restrições (4) e (6) não sejam violadas. A segunda contribuição do trabalho é a implementação de um gerador de instâncias de SIRP, o qual permitirá que algoritmos nesta área sejam comparados.

Dentre os algoritmos implementados, todos apresentaram vantagens e desvantagens. Os experimentos mostraram que o algoritmo Rahim com Simulated Annealing é uma boa alternativa quando desejamos resolver instâncias grandes e complexas. Para instâncias menores o melhor continua sendo o algoritmo original, i.e. o algoritmo Rahim com C&W. A redução do custo em comparação ao algoritmo de Rahim com C&W chega a 17% em média. Essa redução de custo se torna muito significativa quando analisamos a relevância do problema SIRP na operação logística.

Além disso, é possível observar que a diferença entre o tempo computacional para o algoritmo de Rahim com Simulated Annealing e do algoritmo de Rahim com C&W diminui enquanto aumentamos a quantidade de clientes, sendo o algoritmo RSA apenas 10% mais lento para N=50.

O GAP dos resultados confirma a escolha do algoritmo de Rahim com C&W para as instâncias menores e do algoritmo de Rahim com Simulated Annealing para instâncias maiores. Pois mesmo que seja esperado o aumento do valor do GAP conforme o tamanho das instâncias cresça, os melhores GAPs acompanharam os melhores resultados.

Como possíveis trabalhos futuros, sugerimos a adição de diferentes restrições que aproximem o modelo proposto de problemas encontrados por empresas que tenham essas dificuldades. Por exemplo a janela de tempo de atendimento dos clientes, e um nível de confiança individualizado para cada cliente.

Referências

[1] M. K. I. Abdul Rahim, Y. Zhong, E.-H. Aghezzaf, and T. Aouam. Modelling and solving the multiperiod inventory-routing problem with stochastic stationary demand rates. *International Journal of Production Research*, 52(14):4351–4363, 2014.

[2] D. Adelman. A price-directed approach to stochastic inventory/routing. *Operations Research*, 52(4):499–514, 2004.

[3] P. P. Belfiore, O. L. d. V. Costa, and L. P. L. Fávero. Problema de estoque e roteirização: revisão bibliográfica. *Production*, 16:442 – 454, 12 2006.

[4] L. Bertazzi, A. Bosco, F. Guerriero, and D. Lagana. A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, 27:89–107, 2013.

[5] G. R. Bitran and H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1271–81, 1982.

[6] J. Cáceres-Cruz, A. A. Juan, T. Bektas, S. E. Grasman, and J. Faulin. Combining Monte Carlo simulation with heuristics for solving the inventory routing problem with stochastic demands. page 274, 2012.

[7] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[8] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Thirty years of inventory routing. *Transportation Science*, 48(1):1–19, 2013.

[9] R. A. de Carvalho Oliveira, K. V. Delgado, and D. A. Moreira. Sistema para roteamento de veículos capacitados aplicando métodos de Monte Carlo. *iSys-Revista Brasileira de Sistemas de Informação*, 8(3):42–63, 2016.

[10] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management science*, 27(1):1–18, 1981.

[11] H. M. Harmanani, D. Azar, N. Helal, and W. Keirouz. A simulated annealing algorithm for the capacitated vehicle routing problem. In *Proceedings of the ISCA 26th International Conference on Computers and Their Applications*, pages 96–101, 2011.

[12] L. M. Hvattum, A. Løkketangen, and G. Laporte. Scenario tree-based heuristics for stochastic inventory-routing problems. *INFORMS Journal on Computing*, 21(2):268–285, 2009.

[13] A. A. Juan, S. E. Grasman, J. Caceres-Cruz, and T. Bektaş. A simheuristic algorithm for the single-period stochastic inventory-routing problem with stock-outs. *Simulation Modelling Practice and Theory*, 46:40–52, 2014.

[14] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[15] K. Shin and S. Han. A centroid-based heuristic algorithm for the capacitated vehicle routing problem. *Computing and Informatics*, 30(4):721–732, 2012.

[16] N. Shukla, M. Tiwari, and D. Ceglarek. Genetic-algorithms-based algorithm portfolio for inventory routing problem with stochastic demand. *International Journal of Production Research*, 51(1):118–137, 2013.

[17] Y. Yu, H. Chen, and F. Chu. A new model and hybrid approach for large scale inventory routing problems. *European Journal of Operational Research*, 189(3):1022–1040, 2008.

[18] Y. Yu, F. Chu, and H. Chen. A model and algorithm for large scale stochastic inventory routing problem. In *2006 International Conference on Service Systems and Service Management*, volume 1, pages 355–360. IEEE, 2006.