Heurísticas para um Problema Integrado de Sequenciamento da Produção e Roteamento de Veículos de Frota Heterogênea

Gabriel de P. Felix and José E. C. Arroyo

Department of Computer Science, Universidade Federal de Viçosa,
Viçosa - MG, 36570-900, Brazil
gabriel.felix@ufv.br, jarroyo@dpi.ufv.br

Abstract. Este artigo aborda um problema de programação multiobjetivo do sequenciamento de produção de uma máquina linear integrado ao roteamento de veículos para entrega. Nesta abordagem as tarefas possuem penalidades de atraso, datas de vencimento e são caracterizadas por diferentes volumes, com uma frota heterogênea e limitada de veículos (HFVRP). Os critérios a serem minimizados são referentes ao atraso total ponderado e também aos custos de transporte e entrega, incluindo rotas escolhidas e veículos utilizados.
O problema é classificado como NP-Difícil no senso comum, portanto heurísticas eficientes são necessárias para obter soluções próximas da ótima em tempo computacional razoável. Neste trabalho são propostas duas heurísticas baseadas em Iterated Local Search e um Algoritmo Genético (AG). Instâncias para o problema são geradas com base em trabalhos anteriores e os resultados obtidos são comparados aos do solver de otimização CPLEX. Os resultados mostram que as heurísticas aplicadas possuem performance em tempo e qualidade superior às soluções do software a medida em que o número de clientes aumenta.
Keywords: Single machine scheduling, vehicle routing, heterogeneous fleet, time windows, linear programming, meta-heuristics.

1 Introdução

Para aumentar a competitividade no mercado global, as indústrias tem-se concentrado na otimização da cadeia de suprimentos. A cadeia de suprimentos é o conjunto de atividades/processos que envolvem a produção, armazenamento e transporte de produtos ou serviços. Estas atividades incluem a compra de matérias-primas, controle de estoque, produção dos produtos, e entrega dos produtos aos clientes finais dentro dos prazos estabelecidos.

Todas essas atividades devem ser muito bem planejadas e otimizadas para que possam entregar produtos de qualidade e gerar resultados positivos. Os objetivos primordiais que são buscados pela cadeia de suprimentos são: redução de custos, cumprimento de prazos e satisfação dos clientes.

Este trabalho tem como objetivo propor uma abordagem de tomada de decisão para um problema integrado de programação da produção e transporte (roteamento de veículos). Neste problema integrado, um determinado conjunto de tarefas de tamanhos diferentes deve ser processado em uma máquina (disponível numa fábrica) e entregue aos respectivos clientes por uma frota heterogênea de veículos, levando em consideração os prazos de entrega.

O objetivo é minimizar o atraso total ponderado e os custos de transporte (com relação aos custos variáveis dos veículos e custos/tempos de viagem dos veículos). Isoladamente, o problema de agendamento em uma máquina linear com minimização do atraso total é provado como NP-Difícil (Du and Leung, 1990) [1], consequentemente, esta abordagem também é NP-Difícil.

Problemas de programação de produção e roteamento de veículos são dois problemas de otimização combinatória bem estudados na literatura e geralmente são estudados de forma separada.

Pelo conhecimento dos autores deste artigo, o problema integrado de sequenciamento em uma máquina simples e roteamento com veículos de capacidades e custos variáveis, tarefas de diferentes tamanhos, e com o objetivo de minimizar o atraso ponderado total e custos de entrega, ainda não foi abordado na literatura. São componentes deste trabalho:

- Duas abordagens heurísticas de Iterated Local Search (ILS) e um Algoritmo Genético (G.A.).
- Um modelo de programação linear para solução dos problemas integrados.

Na Seção 2 são apresentados estudos prévios na literatura, e na Seção 3 a descrição do problema tratado neste artigo. Em seguida, nas Seções 5 e 5.1 são retratadas a representação de solução escolhida e um caso hipotético do problema, respectivamente. Acompanha esta última um modelo matemático misto de programação linear na Seção 4. Na Seção 6 são descritos parâmetros e restrições utilizadas na geração de instâncias.

Continua...

2 Revisão Literária

São descritos nesta seção estudos prévios realizados em problemas de Agendamento de Produção, Roteamentos de Veículos e a combinação destes dois.

M. Gendreau, G. Laporte and C. Musaragayi (1999) [3] propuseram um algoritmo Tabu Search para o problema de Roteamento de Veículos com Frota Heterogênea (HVRP), e foram obtidas soluções de boa qualidade para as instâncias abordadas. Christian Ullrich (2013) [4] desenvolveu um Algoritmo Genético (A.G.) integrando agendamento de produção de máquinas paralelas e roteamento de uma frota heterogênea, em um ambiente com janelas de tempo de entrega e veículos com diferentes capacidades.

Hall and Potts (2003) [5] abordaram o problema integrado de agendamento de produção e entre de lotes, com vários clientes e manufaturas, e definiram assim a área de Supply Chain Scheduling. Kirlik and Oguz (2012) [6] desenvolveram um General Variable Neighborhood Search (GVNS) para resolver o problema de Minimização do Atraso Total Ponderado com tempos de configuração dependentes da sequência.

Estudos são realizados na área de pesquisa operacional para integrar os dois problemas. Condotta, Knust, Meier, and Shakhlevich (2013) [7] estudaram o problema de minimizar o atraso máximo de uma máquina linear e considerando datas de entrega para os trabalhos, com veículos com mesma capacidade para entrar as tarefas. Há pouco tempo, Hassanzadeh and RastiBarzoki (2017) [8] estudaram a otimização na cadeia de produção considerando o problema de roteamento de veículos com penalidades de atraso e consumo total de recursos.

Tamannaei and Morteza (2019) [9] desenvolveram o primeiro modelo matemático misto de programação linear para o problema de minimização do atraso ponderado total e custos de entrega. Foram propostos três algoritmos genéticos (A.G.) e um método exato Branch-and-Bound (B&B).

3 Descrição do Problema

Nesta seção, o problema discutido neste artigo é apresentado juntamente ao seus parâmetros, índices, variáveis de decisão. Em seguida, a função multiobjetivo tratada no problema é descrita em partes.

Indices:

i, j	Index of jobs.
k	Index of vehicles.
K	Number of vehicles.
N	Number of jobs.

Instance Parameters:

Q_k	Capacity of each vehicle k.
F_k	Cost of each vehicle k.
P_i	Processing time associated to job i.
t_{ij}	Travel time between customer of job i and customer of job j.
w_i	Penalty applied to delivery tardiness of job i.
s_i	Size of job i.
d_i	Due date associated to job i.

Decision Variables:

C_i	Completion time of job i
X_{ij}^k	Binary decision variable which indicates with value '1' that a vehicle k went from customer of job i to customer of job j, and '0' otherwise.
Y_k	Binary variable which represents with value '1' if a vehicle k is used, and '0' otherwise.

S_k Time of a vehicle k starts to be used.
A_{ij} Binary variable which represents if a job i is processed before job j, value '1' if it's true, and '0' otherwise.
D_i Delivery time of job i.
T_i Tardiness of job i.

Seja uma máquina linear que lidará com N tarefas, isto é, cada tarefa i de tamanho s_i deverá ser processada em sequência a partir do início do processo ($t = 0$). Um cliente ordena somente uma tarefa. Caso uma tarefa i seja processada antes de uma tarefa j na máquina, a variável A_ij possuirá valor '1', ou '0' caso contrário. O tempo de completude de processamento para tarefa i é representado por C_i. Para a entrega das tarefas, são disponibilizados K veículos, realizando circuitos partindo da origem 'O', sendo a distância entre dois clientes i e j representada por t_ij. Na rota percorrida para entrega, caso uma tarefa i seja entregue imediatamente anterior à uma tarefa j e seja entregue por um veículo k, esta será representada pela variável X_{ij}^k com valor '1', ou '0' caso contrário. A distância de trajeto entre dois clientes i e j é representada por t_{ij}.

Cada veículo k possui capacidade Q_k e custo de utilização F_k, além de possuir tempo de ínicio S_k. Caso um veículo k seja utilizado, será representado em Y_k com o valor '1', ou '0' caso contrário.

Cada tarefa i possui tempo de processamento P_i e data de vencimento d_i vinculada, que pode ser penalizada por uma constante w_i multiplicada pelo seu atraso T_i. O tempo de chegada no destinatário é representado por D_i. A variável de atraso T é dada por $D_i - d_i$, i.e., a diferença entre o tempo de chegada e a data de vencimento.

A minimização do trajeto de entrega, custos utilização de veículos e atraso ponderado total é representada pela função objetivo:

$$\min \sum_{i,j=0}^{N} \sum_{k=0}^{K} X_{ij}^k * t_{ij} + \sum_{k=0}^{K} F_k * Y_k + \sum_{i=1}^{N} w_i * T_i \tag{1}$$

4 Mathematical Model

Nesta seção, um modelo matemático linear misto é apresentado. Este é modelo é uma versão modificada do proposto por (Tamannaei and Rasti-Barzoki, 2019) [9], sendo o primeiro desenvolvido na versão do problema em que veículos possuem mesma capacidades (Q) e custos (F). O modelo é descrito a seguir:

$$\min \sum_{i,j=0}^{N} \sum_{k=0}^{K} X_{ij}^k * t_{ij} + \sum_{k=0}^{K} F_k * Y_k + \sum_{i=1}^{N} w_i * T_i \tag{1}$$

st.

$$\sum_{k=0}^{K} \sum_{j=0, j \neq i}^{N} X_{ij}^{k} = 1, \quad \forall i = 1 \dots N \tag{2}$$

$$\sum_{i=0}^{N} \sum_{j=0, j \neq i}^{N} s_i * X_{ij}^{k} \leq Q_k * Y_k, \quad \forall k = 0 \dots K; \tag{3}$$

$$\sum_{j=0}^{N} X_{0j}^{k} = Y_k, \quad \forall k = 0 \dots K; \tag{4}$$

$$\sum_{i=0}^{N} X_{ih}^{k} = \sum_{j=0}^{N} X_{hj}^{k}; \quad \forall h = 0 \dots N; \forall k = 0 \dots K \tag{5}$$

$$A_{ij} + A_{ji} = 1; \quad \forall i, j = 0 \dots N; i \neq j; \tag{6}$$

$$A_{ij} + A_{jr} + A_{ri} \geq 1; \quad \forall i, j, r = 0 \dots N; i \neq j \neq r; \tag{7}$$

$$C_j = \sum_{i=1, i \neq j}^{N} (P_i * A_{ij}) + P_j; \quad \forall j = 1 \dots N; \tag{8}$$

$$S_k \geq C_j - M * (1 - \sum_{i=1, i \neq j}^{N} X_{ij}^{k}); \quad \forall j = 1 \dots N; \forall k = 0 \dots K; \tag{9}$$

$$D_j - S_k \geq t_{0j} - M * (1 - X_{0j}^{k}); \quad \forall k = 0 \dots K; \forall j = 1 \dots N \tag{10}$$

$$D_j - D_i \geq t_{ij} - M * (1 - \sum_{i=1}^{N} X_{ij}^{k}); \quad \forall i, j = 1 \dots N; \tag{11}$$

$$T_i \geq D_i - d_i; \quad \forall i = 1 \dots N; \tag{12}$$

$$C_0 = 0; \tag{13}$$

$$D_0 = 0; \tag{14}$$

$$T_0 = 0; \tag{15}$$

$$X_{ijk} \in \{0, 1\}; \quad \forall i, j = 0 \dots N, \forall k = 0 \dots K; \tag{16}$$

$$A_{ij} \in \{0, 1\}; \quad \forall i, j = 1 \dots N; \tag{17}$$

$$Y_k \in \{0, 1\}; \quad \forall k = 0 \dots N; \tag{18}$$

$$S_k \geq 0; \quad \forall k = 0 \dots K; \tag{19}$$

$$T_i \geq 0; \quad \forall i = 1 \ldots N; \tag{20}$$

$$D_i \geq 0; \quad \forall i = 1 \ldots N; \tag{21}$$

The objective function (1) is to minimize the travel distance, total tardiness weight of the jobs and the used vehicles costs. Constraints (2) ensure that each job is carried by exactly one vehicle.

Constraints (3) guarantee that sum of jobs in each vehicle doesn't overcome it's capacity. Constraints (4) indicates that if a vehicle is in use, it will leave the origin.

Constraints (5) ensure that if a vehicle arrives at a customer, it will also leave that customer. Constraints (6) and (7) are to maintenance the processing order, demanding jobs to be processed in a sequential and possible order.

Constraints (8) defines the completion time of a job as the sum of previous jobs' processing times and its own. Constraints (9) guarantee that if a job j is carried by a vehicle k, the vehicle start time will be greather than job j's completion time.

Constraints (10) indicates that if a job j is carried by a vehicle k is the first of its tour, its destination time D_j is greater or equal than the sum of start vehicle time S_k and distance between origin and this customer Constraints (11) certify that if a job j delivered by a vehicle k isn't the first of its tour, its delivery time will be greater or equal than the sum of the previous job i delivery time D_i and the start vehicle time S_k.

Constraints (12) defines the tardiness T of a job as the difference between its destination time (D) and its due date (d). Constraints (13), (14) and (15) certifies that the origin (customer zero) do not assume any impossible values.

Constraints (16), (17) and (18) are to define these variables as binary variables. Constraints (19), (20) and (21) ensure that these variables won't assume negative values.

5 Representação de Solução

A representação de uma solução é disposta linearmente, de modo que tarefas e veículos sigam acompanhados de seus respectivos códigos de identificação. O conjunto de tarefas N do problema será dividido em K subconjuntos, sendo cada subconjunto destinado à um veículo específico. O tempo de ínicio de funcionamento de um veículo k, denotado por S_k, é dado pela soma dos tempos de processamento P destas tarefas somado ao tempo acumulado das tarefas processadas anteriormente (a partir de $t = 0$). Em cada subconjunto há definição de uma rota, que indica o trajeto percorrido por um veículo. Seja R uma solução viável, $k \in K$ um veículo e R_k sua rota, a representação deste trajeto é dada por:

$$R_k = O \rightarrow J_1 \rightarrow J_2 \rightarrow ... \rightarrow J_{|N_k|} \rightarrow O \tag{22}$$

Neste contexto, Ji são as tarefas entregues no trajeto e $\forall i = 1, ...|N_k| \in N$ e $|N_k|$ a quantidade de tarefas dispostas neste veículo. Todo veículo parte da origem, e deve retornar para ela, visitando cada cliente somente uma vez. A Figura 1 representa visualmente uma solução.

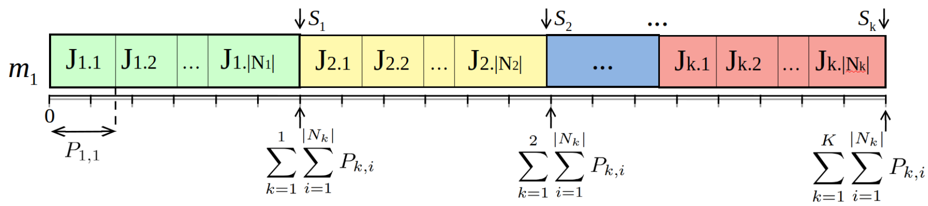


Fig. 1. Representação de solução com tarefas organizadas em lotes.

Um subconjunto de tarefas pode não possuir tarefas e ser designado à um veículo w, indicando que este não está em uso e portanto $|N_w| = 0$.

5.1 Exemplo Numérico

Um caso hipotético de rede de transporte é descrito abaixo, no qual a origem e seis clientes (C_1 à C_6) são dispostos cartesianamente, sendo a distância entre eles representada pelas distâncias euclidianas dos pontos (Tabela 1).

Table 1. Euclidian distance (ED) between two points in the network.

ED(x,y)	Origin	C_1	C_2	C_3	C_4	C_5	C_6
Origin	0	27	260	253	254	112	90
C_1	27	0	265	270	237	114	105
C_2	260	265	0	474	212	371	175
C_3	253	270	474	0	507	178	307
C_4	254	237	212	507	0	346	231
C_5	112	114	371	178	346	0	198
C_6	90	105	175	307	231	198	0

As tarefas e a frota de veículos são detalhadamente descritos na Tabela 2. As informações sobre as tarefas definem a prioridade e urgência de suas entregas. Três veículos heterogêneos(V_1, V_2 e V_3) são disponibilizados para transportá-las.

A Figura 2 descreve visualmente a solução ótima para a instância. Neste exemplo, todos os três veículos foram utilizados durante a entrega, possuindo tempos de início S_1,S_2 e S_3 iguais a 180, 301 e 88, respectivamente. A porcentagem no final de um trajeto representa o espaço ocupado pelas tarefas carregadas no veículo. Dentre as seis tarefas, apenas J_6 foi entregue sem atrasos.

Na solução ótima da instância, os tempos de completude C das tarefas são, respectivamente: 276, 180, 88, 301, 52 e 137.

Table 2. Jobs and vehicles detailed informations.

Job	P_i	d_i	w_i	s_i	Vehicle	Q_k	F_k
J_1	25	266	1.8	20	V_1	204	1224
J_2	49	347	2.3	31	V_2	186	1116
J_3	36	303	4.3	25	V_3	160	960
J_4	96	431	4	86			
J_5	52	177	2	33			
J_6	43	315	4.6	42			

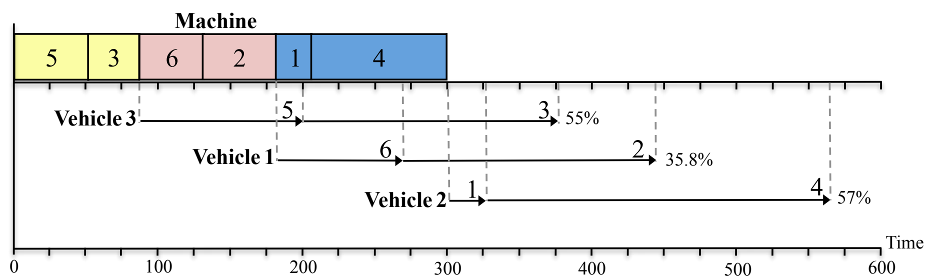


Fig. 2. Gráfico de Gantt com a descrição visual da solução.

Os tempos de destino D, que indicam o momento em que o pedido chega àos clientes são: 328, 445, 378, 565, 200 e 270.

No cálculo da diferença entre o tempo de destino e a o prazo de entrega, obtêm-se os seguintes valores de atraso T: 62, 98, 75, 134, 23 e 0.

O resultado ótimo da função objetivo possui valor absoluto 6127.5,sendo composto por três partes integrantes do problema. A primeira, é dada pelos custos de trajeto dos três veículos, que são calculados em $DC = (112 + 178 + 253) + (90 + 175 + 260) + (27 + 237 + 254) = 1586$.

A segunda parte é definida pelos custos de utilização de veículo, que neste contexto será total, uma vez que todos veículos foram utilizados. Desta forma, $VC = 1224 + 1116 + 960 = 3300$.

Os atrasos T serão multiplicados por suas respectivas penalidades w. Isto posto, o custo total de atrasos ponderados é $TC = (62 * 1.8) + (98 * 2.3) + (75 * 4.3) + (134 * 4) + (23 * 2) + (0 * 4.6) = 1241.5$.

Portanto, ao combinar as três parcelas da função objetivo, é obtido o valor final $DC + VC + TC = 6127.5$ na solução ótima da instância.

6 Geração de Instâncias

Por se tratar de uma variação do problema ainda não abordada na literatura, a criação de instâncias tornou-se necessária. As instâncias foram distruibuídas em dois grupos, instâncias pequenas e grandes. O primeiro varia entre oito e vinte tarefas, de três à seis veículos, e o segundo entre trinta e cem tarefas, de cinco à doze veículos.

Seja uma tarefa $i \in N$, seu tempo de processamento P_i pertence um intervalo limitado:

$$P_i \in rand[1, \rho], \quad with \quad \rho = 100 \tag{1}$$

Seu tamanho s_i é relacionado ao tempo de processamento, também definido em um intervalo:

$$s_i \in rand[1, P_i] \tag{2}$$

A data de vencimento d_i de uma tarefa deve possuir ligação entre seu tempo de processamento e a distância entre o cliente e a origem. Ela é determinada a partir de uma janela de tempo, que possui limite inferior z_i e superior $\overline{z}_i$:

$$\begin{aligned} d_i \in rand[z_i, \overline{z}_i] \quad st. \\ z_i = P_i + t_{0i} + \pi_1 \\ \overline{z}_i = z_i + \pi_2 \end{aligned} \tag{3}$$

Os parâmetros π_1 e π_2 influenciam na dificuldade da instância pois seus valores podem encurtar ou prolongar as janelas de tempo, resultando em prazos de entrega menores ou maiores:

$$\begin{aligned} &\pi_1 \in rand[0, \tfrac{\sum_{i=1}^{N} P_i}{K+1}] \\ &\pi_2 \in rand[0, \delta * \rho] \end{aligned} \tag{4}$$

O peso de penalidade w_i é determinado de forma aleatória em um intervalo, sendo este parâmetro integrante da função objetivo:

$$w_i \in rand[1.0, 5.0] \tag{5}$$

A capacidade Q dos veículos é dada de forma que a tarefa de maior tamanho possa ser transportada qualquer um destes, somado à um valor relacionado a média de todos tamanhos pela quantidade de veículos:

$$\begin{aligned} &Q_k = max_{i=1}^{N}\{s_i\} + \tau \quad st. \\ &\tau \in rand[\tfrac{\sum_{i=1}^{N} s_i}{K}, \mu * \tfrac{\sum_{i=1}^{N} s_i}{K}] \end{aligned} \tag{6}$$

O custo F_k de um veículo k é diretamente proporcional à sua capacidade:

$$F_k = N * Q_k \tag{7}$$

To determine the travel time t_{ij} between two customers, first we have to generate the origin's coordinates. The machine's coordinates (x_0, y_0) are generated subject to the following range:

$$x_0, y_0 \in rand[400, 700] \tag{8}$$

Após fixadas as coordenadas da origem. são geradas as coordenadas dos clientes de forma que estejam no primeiro quadrante e sua distância euclidiana (ED) até a origem respeite a seguinte constante:

$$\begin{aligned}&(x_i, y_i) \geq (0,0) \quad \forall i = 1, ..., N \\ &ED((x_0, y_0),(x_i, y_i)) \leq (\rho * K) \quad \forall i = 1, ..., N\end{aligned} \tag{9}$$

Por fim, são indicados os parâmetros utilizados nas equações acima (δ,μ, N and K). Para cada um dos conjuntos de instâncias (small e large), existe uma definição diferente dos valores dos parâmetros, que influenciará na dificuldade da resolução do problema.

Para instâncias pequenas, os valores definidos para μ, N and K são:

$$\begin{aligned}&\mu \in \{1, 1.5, 2.0\} \\ &N \in \{8, 10, 15, 20\} \\ &K \in \{3, 4, 5, 6\}\end{aligned} \tag{10}$$

Para instâncias maiores, após realizados os ajustes dos três parâmetros, estes foram definidos em:

$$\begin{aligned}&\mu \in \{1.5, 2.0, 2.5\} \\ &N \in \{50, 80, 100\} \\ &K \in \{5, 8, 10, 12\}\end{aligned} \tag{11}$$

A constante δ utilizada para calcular o limite superior de d_i (Equação 4), assume os seguintes valores para ambos conjuntos de instâncias:

$$\delta \in \{0.5, 1.0, 1.5, 2.0, 2.5\} \tag{12}$$

Definidos os parâmetros, para cada combinação (δ,μ, N and K) foram geradas cinco instâncias pequenas e três instâncias grandes, totalizando (5*3*4*4*5) + (3*3*3*4*5) = 1740 instâncias.

7 Abordagem de Solução

São tratadas no artigo três abordagens para solucionar o Problema de Sequenciamento de Produção e Entrega, sendo duas versões da metaheurística Iterated Local Search (ILS) e um Algoritmo Genético (G.A.). Ullrich (2013) [4] atingiu bons resultados com a aplicação de um Algoritmo Genético para o ambiente de máquinas paralelas do problema.

O método de busca local aplicado junto aos algoritmos é o procedimento VND (Mladenovic and Hansen 1997) [11], com ordenação aleatória de vizinhanças (RVND). São descritas duas versões de RVND neste trabalho. Em uma das versões, uma técnica de aprimoramento interno de rotas é aplicada.

A estrutura de ambas Iterated Local Search são similares, diferenciando-se na chamada dos métodos de busca local RVND. O Algoritmo Genético (G.A.) desenvolvido utiliza técnicas de Torneio Binário e 2-point Crossover.

7.1 Soluções Iniciais

A qualidade de um ótimo local obtido por método de busca local depende da solução inicial trabalhada (El-Ghazali Talbi, 2009) [12]. As regras para a criação de soluções iniciais utilizadas aqui originam-se do problema de Permutation Flow Shop, sendo estas Apparent Tardiness Cost (ATC), Weighted Modified Due Date (WMDD) e Weighted Earliest Due Date (EDD) (Molina-Sánchez and González-Neira, 2015) [13].

O conjunto de regras têm por objetivo designar uma ordem favorável para o sequenciamento de tarefas. Por se tratar de uma Frota Heterogênea de Veículos, a ordem gerada pelas regras pode ou não ser atendida devido às restrições de capacidade dos veículos. Caso nenhuma das três regras gere sequência viável, uma solução inicial aleatória é utilizada.

7.2 Operadores de Vizinhança

O processo de busca local visa aprimorar uma solução atual ao considerar soluções similares pelas operações de vizinhança definidas. Neste trabalho, são consideradas apenas soluções viáveis no processo. Portanto, todas as soluções em que a soma dos tamanhos das tarefas exceda a capacidade de transporte de um veículo serão desconsideradas. Matematicamente, uma solução S é descartada se:

$$Q_k < \sum_{i=1}^{|N_k|} s_i; \forall k \in K; \tag{13}$$

Para constituir o conjunto de vizinhanças de uma solução, estas foram subdivididas em dois subconjuntos: vizinhanças Intra-Rota e Inter-Rota.

7.3 Vizinhanças Intra-Rota

Neste subconjunto, as operações aplicadas para diversificação da solução são restritas a alterações internas de um veículo, influenciando somente na ordem de entrega das tarefas carregadas por este. Seja R uma solução viável para o problema e R_i a rota completa de um veículo i. São definidas as operações:

– **Swap-Adj-Job**(R_i) — Troca de posição duas tarefas adjacentes na rota R_i.

– **Insert-Job**(R_i) — Insere uma tarefa nas demais posições da rota R_i.

– **2-opt**(R_i) — Realiza a operação de 2-opt em todas combinações de arcos distintos da rota R_i, i.e., seleciona dois arcos distintos, os remove e reconecta os clientes de forma que a rota seja modificada e permaneça interligada.

7.4 Vizinhanças Inter-Rota

As operações de vizinhança do tipo Inter-Rota consideram duas ou mais rotas durante sua aplicação, realizando alterações em mais de uma rota ao mesmo tempo. Durante o processo, as operações aplicadas podem gerar soluções inviáveis, porém estas não serão consideradas neste trabalho.

Seja R uma solução viável para o problema e R_i e R_w as rotas completas realizadas pelos veículos $i,w \in K$. Ademais, o conjunto de tarefas carregadas por um veículo i é representado por N_i. São definidas as operações:

- **Swap-Job**(R_i, R_w) — Trocam de posição duas tarefas $a \in N_i$ e $b \in N_w$ entre as duas rotas, de forma que a receberá a posição antiga de b na nova rota e vice-versa.

- **Insert-Job**(R_i, R_w) — Insere uma tarefa $a \in N_i$ nas demais posições da rota R_w.

- **Swap-Adj-Vehicles**(R_i, R_w) — Trocam de posição dois veículos adjacentes i e w carregando consigo seus lotes de tarefas, i.e., caso o veículo i agora parta após o veículo w, o primeiro (i) deverá esperar todas as tarefas do segundo (w) sejam processadas para que as suas se iniciem na máquina.

- **Insert-Vehicle**(R_i) — Insere um veículo i juntamente com suas tarefas atreladas (N_i) nas demais posições da solução, i.e., insere seu lote de tarefas antes, entre ou após outros veículos.

7.5 Abordagens de RVND

O processo de busca local é realizado por VND (Mladenovic and Hansen 1997) [11], utilizando sete operadores de vizinhança e ordenação aleatória de vizinhanças (RVND). São empregadas duas versões de RVND, com técnicas diferentes de organização dos operadores.

A primeira versão, descrita em Algorithm 1, consiste de percorrer aleatoriamente pelo conjunto de vizinhanças à procura de melhorias na solução atual, e em seguida reiniciar este procedimento quando alguma melhora é alcançada. Do contrário, caso nenhuma melhoria seja obtida na execução, a busca local é encerrada. O algoritmo possui um único parâmetro: uma solução S, e se inicia atribuindo S à melhor solução encontrada $S*$ (Passo 1).

Em seguida, todo o conjunto de vizinhanças é distribuido em ordem aleatória em $NeighborhoodSet$ (Passo 2). Todas as vizinhanças em $NeighborhoodSet$ são visitadas na ordem sorteada (Passos 3-10) e seus vizinhos S são comparados à melhor solução $S*$ (Passos 5-6). Caso algum vizinho seja satisfatório, ele é então salvo em $S*$ (Passo 7) e o processo de busca é reiniciado com nova ordem aleatória (Passos 8-9), caso contrário o algoritmo termina. A melhor solução $S*$ encontrada na busca local é retornada.

Algorithm 1: RVND(S)

```
Input    : Solution S
Output : Better or equal S* solution
S* ← S;
NeighborhoodSet ←Random Order(Neighborhoods);
for i ← 1 in NeighborhoodSet do
    currNeighborhood ← NeighborhoodSet[i]; // Select Neighborhood
    S ← currNeighborhood(S); // Apply Local Search
    if f(S) ≤ f(S*) then
        S* ← S;
        NeighborhoodSet ←Random Order(Neighborhoods);
        j ← 1; // Restart Local Search
    end
end
Output : S*
```

A segunda versão, RVND-Custom, é baseada na técnica de aprimoramento interno (Penna and Ochi, 2013) [14], onde as vizinhanças externas à uma rota Inter-Route Neighborhoods são analisadas, e caso seja obtida melhora na solução, as vizinhanças internas Intra-Route Neighborhoods são acionadas para a aprimorar ainda mais. A aplicação é descrita em Algorithm 2.

O algoritmo possui um parâmetro: uma solução S, e é iniciado ao atribuir S à melhor solução encontrada $S*$ (Passo 1). Em seguida, o conjunto de vizinhanças Inter-Rota é randomicamente distribuido e armazendo em $InterRoute$ (Passo 2). Todos as vizinhanças são visitadas na ordem sorteada (Passos 3-22), sendo seus vizinhos S' analisados (Passos 4-5) e caso sejam melhores do que S, o aprimoramento interno de rota se inicia (Passos 6-18).

Em seguida, o conjunto de vizinhanças Intra-Rota é visitado sequencialmente (Passos 8-15) e na hipótese de melhorar novamente (Passos 11-14), a busca interna se reinicia (Passo 13). Do contrário, esta termina e a busca externa é reiniciada com nova ordem aleatória (Passos 16-17). A melhor solução encontrada é atualizada a cada iteração da busca externa (Passos 19-21). Ao final do algoritmo, a melhor solução $S*$ é retornada.

8 Iterated Local Search

Ao gerar variados ótimos locais, utilizar a Iterated Local Search torna possível obter aprimoramento da qualidade da solução (Talbi, 2009) [12]. Neste artigo em específico, as aplicações de Iterated Local Search utilizam duas versões de busca local RVND.

Os métodos de perturbação do algoritmos descritos em seguida utilizam os operadores de vizinhança Inter-Rota: **Swap-Job**(R_i, R_w) e **Insert-Job**(R_i). É realizada uma quantidade α de perturbações, sendo α estatisticamente definido na Seção 'Calibração de Parâmetros'.

Algorithm 2: RVND-Custom(S)

```
Input   : Solution S
Output : Better or equal S* solution
S* ← S;
InterRoute ←Random Order(Inter-Route Neighborhoods);
for i ← 1 in InterRoute do
    currInter ← InterRoute[i]; // Get Inter-Route Neighborhood
    S′ ← currInter(S); // Apply Inter Local Search
    if f(S′) ≤ f(S) then
        IntraRoute ←Intra-Route Neighborhoods;
        for j ← 1 in IntraRoute do
            currIntra ← IntraRoute[j];// Apply Intra Local Search
            S″ ← currIntra(S′);
            if f(S″) ≤ f(S′) then
                S′ ← S″;
                j ← 1; // Restart Intra-Route Socal Search
            end
        end
        i ← 1;
        InterRoute ←Random Order(Inter-Route Neighborhoods);
    end
    if f(S′) ≤ f(S*) then
        S* ← S′;
    end
end
Output : S*
```

8.1 ILS RVND

O esquema geral da metaheurística ILS proposta é apresentado em Algoritmo 3. Existem três parâmetros: α (número de perturbações), $MaxIter$ (quantidade de reinícios de solução) e $MaxIterILS$ (máximo de iterações consecutivas para o algoritmo). Inicialmente, a variável de melhor solução global é inicializada (Passo 1).

A cada iteração externa, uma nova solução inicial S é gerada (Passo 3) utilizando as regras descritas na Seção 7.1, e em seguida é feito busca local em S (Passo 4). Após sua possível melhora, o processo de Perturbação é inicializado (Passos 5-12), a solução atual S' é perturbada com intensidade α (Passo 10), é feito busca local RVND (Passo 11) e a melhor solução encontrada é atualizada (Passos 6-9).

Algorithm 3: ILS-RVND ($\alpha, MaxIter, MaxIterILS$)

Input : Parameters of max. iterations numbers
Output : Best solution $S*$
1 $S* \leftarrow$ Initialize($S*$);
2 **for** $i \leftarrow 1$ to $MaxIter$ **do**
3     $S \leftarrow$ GenerateInitialSolution();
4     $S' \leftarrow$ RVND(s);
5   **for** $j \leftarrow 1$ to $MaxIterILS$ **do**
6   **if** f(S) $\leq$ f(S') **then**
7             $S' \leftarrow S$;
8             $j \leftarrow 1$; `// Restart Perturbing Process`
9   **end**
10         $S' \leftarrow$Perturb(S', α);
11         $S' \leftarrow$RVND(S'); `// Local Search on Perturbed S'`
12   **end**
13   **if** f(S') $\leq$ f($S*$) **then**
14         $S* \leftarrow S'$;
15   **end**
16 **end**
Output : $S*$

Caso alguma melhora em S' em relação à solução inicial S seja obtida, o processo de Perturbação é reinicializado (Passo 8). Nos Passos 13-15, S' é comparada a melhor solução $S*$ e finaliza uma iteração do algoritmo. Ao final é retornada a melhor solução encontrada $S*$.

8.2 ILS-RVND com Aprimoramento de Rota Interna

Penna and Ochi, 2013 [14] desenvolveram o primeiro modelo de Iterated Local Search com RVND para o Problema de Roteamento de Frota Heterogênea de Veículos (HFVRP) em um ambiente com até cem clientes, e produziram resultados competitivos com aos da literatura. Sua abordagem em lidar com a entrega de mercadorias em específico serviu de inspiração para esta combinação ILS-RVND.

A diferença desta versão de Iterated Local Search para a anterior está na busca local RVND-Custom, que utiliza a técnica de aprimoramento interno de rotas. O pseudocódigo desta aplicação é derivado do Algoritmo 3 ao realizar a substituição referida em Figura 3.

4,11 $S' \leftarrow$ RVND(s);

4,11 $S' \leftarrow$ RVND-Custom(S);

Fig. 3. Mudança entre acionamentos de RVND.

Nesta técnica, assim que uma solução atual é aperfeiçoada utilizando operadores externos às rotas, como realizar intercâmbio de tarefas entre veículos, os operadores internos às rotas são acionados.

9 Genetic Algorithm

Algoritmos Genéticos foram desenvolvidos por J. Holland em meados dos anos 70 (University of Michigan, USA) para entender o processo adaptativo de sistemas naturais (Talbi, 2009) [12]. Nesta abordagem, são utilizadas técnicas de Torneio Binário para seleção de soluções e 2-point Crossovers para diversificação da prole. O esquema geral do algoritmo é descrito em Algorithm 4.

Existem dois parâmetros: $PopSize$ (quantidade de indivíduos na população) e $MaxIter$ (número de iterações em que serão geradas novas populações. O algoritmo é iniciado no Passo 1 ao gerar uma população aleatória P, seguido de uma busca local RVND em P (Passo 2). A melhor solução da população é armazenada em $P*$ (Passo 3).

A cada iteração uma nova solução é inserida em P' até o limite de $PopSize-1$ indivíduos (Passos 6-22). Para gerar uma prole, são escolhidas duas soluções 'pai' por Torneios Binários (Passos 7-8), e em seguida são geradas duas mutações destas soluções (Passos 9-10).

O processo de mutação Crossover pode produzir soluções inviáveis, dessa forma a escolha de uma nova solução para a população P' necessita de análise:

- No primeiro caso, em que as duas novas mutações são viáveis, estas participam de um Torneio Binário (Passo 12) e a escolhida será adicionada à população.
- No segundo caso, onde apenas uma das soluções é viável, a única prole viável será inserida na população P' (Passo 15).

Algorithm 4: Genetic Algorithm ($PopSize, MaxIter$)

```
Input   : Population size and max. iteration number
Output  : Best solution P* in MaxIter populations
P ← Generate random population(PopSize);
P ← RVND(P); // Apply local search to new population P
P* ← Best(P); // Save best solution of P
for i ← 1 to MaxIter do
    P' ← Ø;
    for j ← 1 to (PopSize − 1) do
        Parent1 ← Binary Tournament(random(P), random(P));
        Parent2 ← Binary Tournament(random(P), random(P));
        Offspring1 ← Crossover(first, second);
        Offspring2 ← Crossover(first, second);
        if Feasible(Offspring1) and Feasible(Offspring2) then
            P' ← P' + BinaryTournament(Offspring1, Offspring2);
        else
            if Feasible(Offspring1) or Feasible(Offspring2) then
                P' ← P' + Feasible from(Offspring1, Offspring2);
            else
                if !Feasible(Offspring1) and !Feasible(Offspring2) then
                    P' ← P' + Mutation(random(Parent1, Parent2));
                end
            end
        end
    end
    P' ← P' + P*; // Include best solution to P'
    P' ← Fast Local Search(P');
    if f(P') ≤ f(Best(P')) then
        P' ← P*;
    end
    P ← P';
end
Output : P*
```

– No último caso, se as duas mutações geradas são inviáveis (Passo 17), uma escolha aleatória entre as soluções ‘pai’ é feita e sofrerá uma nova mutação Crossover (Passo 18), sendo adicionada em seguida.

Após a obtenção de $PopSize - 1$ indivíduos em P', é inserida a melhor solução $P*$ encontrada até o momento (Passo 23). Com a população completa, é realizada uma busca local rápida (Passo 24) em todos os indivíduos. Este método é descrito na Seção 5. Por fim, a melhor solução $S*$ é atualizada (Passos 25-26) e a população P' torna-se a população atual P, para nas próximas iterações gerar novos indivíduos (Passo 25).

9.1 Fast Local Search

Nesta seção é descrito um procedimento de busca local chamado Fast Local Search. Métodos de busca local convencionais como VND e RVND podem ser computacionalmente custosos acionados consecutivas vezes, apesar de possibilitarem melhoras significativas na qualidade da solução. Dessa forma, foi implementado um método de busca local simples para lidar com as soluções do algoritmo populacional. O procedimento é descrito em Algoritmo 5.

Algorithm 5: Fast Local Search(S)

Input : Solution S
Output : Better or equal $S*$ solution
1 $S* \leftarrow S$;
2 $NeighborhoodSet \leftarrow$ *Inter-Route* and *Intra-Route Neighborhoods*;
3 **for** $i \leftarrow 1$ to $NeighborhoodSet$ **do**
4     $currNeighborhood \leftarrow NeighborhoodSet[i]$;
5     $S' \leftarrow$ currNeighborhood(S);
6   **if** f(S') $\leq$ f($S*$) **then**
7         $S* \leftarrow S'$;
8   **end**
9 **end**
Output : $S*$

O algoritmo possui um parâmetro: uma solução S e se inicia ao atribuir S à melhor solução global $S*$ (Passo 1). Todo o conjunto de vizinhanças (Inter-Rota e Intra-Rota) é atribuido à $NeighborhoodSet$ (Passo 2). Em seguida, o método percorre sequencialmente as vizinhanças em $NeighborhoodSet$ (Passos 3-8) e analisa seus vizinhos (Passos 4-5). Em cada vizinhança, a melhor solução S' da vizinhança é comparada com a melhor solução global e $S*$ é atualizada (Passos 6-8). Por fim, o método de busca local retorna a melhor solução encontrada.

10 Experimentos Computacionais

Nesta Seção são apresentados os experimentos realizados em cima dos três algoritmos e o software CPLEX. Todos os algoritmos foram codificados em C++ e executados em uma máquina Intel(R) Core TM i7-4790K CPU @ 4.00GHz x 8 with 32GB RAM, rodando Ubuntu ??.?? 64 bits.

A métrica de comparação utilizada é o desvio relativo percentual (RPD), sendo utilizada a melhor média entre os algoritmos (f_{best}) para a comparação entre elas. O RPD mede percentualmente o quanto os resultados experimentais (f_{method}) diferem do valor de referência:

$$RPD = \frac{f_{method} - f_{best}}{f_{best}} * 100\% \tag{14}$$

A performance das heurísticas aplicadas foram testadas em todas 1740 instâncias, sendo cada heurística executada cinco vezes em cada instância. A média entre das execuções é considerada para as comparações. O software CPLEX foi executado uma vez para cada instância pequena, onde $N \in \{8, 10, 15, 20\}$ e $V \in \{3, 4, 5, 6\}$.

As siglas para as heurísticas nesta seção foram definidas de forma que 'ILS-RVND-1' se refira à heurística ILS RVND, 'ILS-RVND-2' à heurística ILS RVND com Aprimoramento de Rota Interna e por fim 'GA-LS' ao Algoritmo Genético.

10.1 Calibração de parâmetros dos algoritmos

Nesta Seção é apresentado o método e os parâmetros determinados para execução dos três algoritmos. Testes preliminares foram realizados para definir os parâmetros dos algoritmos.

O primeiro, ILS_RVND_1, utiliza três parâmetros para sua execução: a intensidade de perturbação α, o número máximo de reinícios do algoritmo $MaxIter$ e o máximo de iterações da ILS $MaxIterILS$. Foram definidos no conjunto de testes os seguintes valores: $\alpha \in \{2, 5, 8, 10\}$, $MaxIter \in \{5, 8, 10\}$ e $MaxIterILS \in \{50, 100\}$. Dessa forma, foram geradas 24 combinações de parâmetros, e cada uma delas foi executada 10 vezes para as 1740 instâncias.

Os resultados foram analisados pelo teste não-paramétrico de Kruskall-Wallis utilizando o RPD como variável de resposta. Seguindo a equação 14, f_{method} foi definido como sendo o resultado da combinação do algoritmo para aquela instância e f_{best} como o melhor resultado obtido dentre todas combinações. No teste de Kruskall-Wallis duas hipóteses são testadas: a hipótese nula afirma que a performance de todas implementações testadas são estatisticamente similares, e a hipótese alternativa conclui que pelo menos uma implementação possui performance significativamente diferente. Na Figura 4, os intervalos médios de confiança são apresentados à 95% de confiança.

Intervalos que se sobrepõe sugerem que pode não haver diferença entre as implementações. Os intervalos '8-100-5' e '10-100-5' se subreposeram se apresentaram com menor média. Pelo teste de Kruskall-Wallis, com p-valor $0.00 < 0.05$ indica que a hipótese nula deve ser rejeitada, concluindo que pelo menos

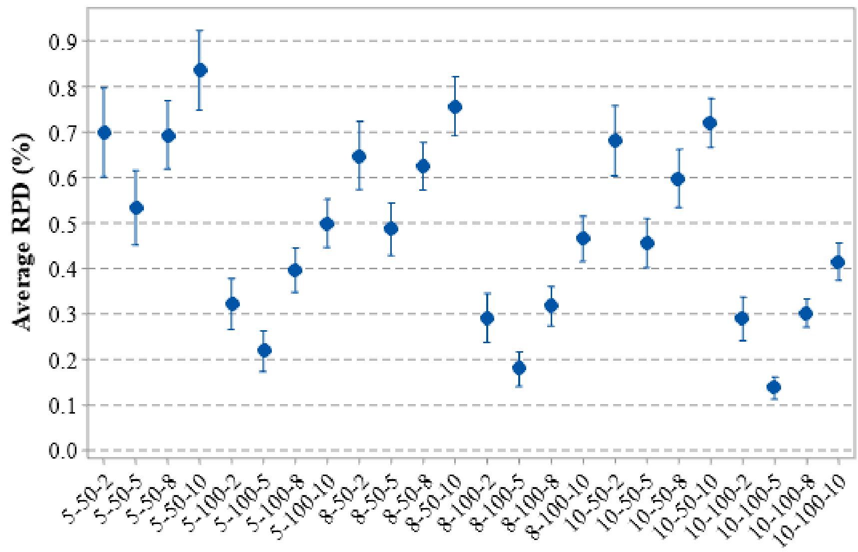


Fig. 4. Mean plot and confidence intervals at 95% confidence level for ILS_RVND_1 experiment.

uma implentação tem performance significamente diferente (hipótese alternativa). Dessa forma, a combinação '10-100-5' foi escolhida por apresentar melhor performance.

De forma análoga, para o segundo algoritmo ILS_RVND_2, foram utilizadas as 24 combinações acima e assim executadas 10 vezes em cada uma das instâncias. As combinações que possuiam menor intensidade de perturbação α obtiveram destaque (Figura 5.

As duas melhores médias '8-100-2' e '10-100-2' não se sobrepuseram nos intervalos de confiança, indicando que há diferença entre as implementações. Ademais, com p-valor $0.00 < 0.05$, a hipótese alternativa é novamente válida e há pelo menos uma implementação com performance significamente diferente. Isto posto, foi definida a combinação '10-100-2' para o algoritmo, por possuir menor média.

O terceiro algoritmo, GA_LS, utiliza dois parâmetros: $PopSize$ e $MaxIter$, o tamanho da população e o número máximo de iterações, respectivamente. Os conjuntos para estes parâmetros foram definidos em $PopSize \in \{30, 50, 70\}$ e $MaxIter \in \{5, 8, 10\}$. Foram geradas 9 combinações de parâmetros e foram executadas sobre as 1740 instâncias.

Novamente, pelo teste de Kruskall-Wallis, o p-valor $0.00 < 0.05$ indica a validade da hipótese alternativa. Os intervalos de confiança são mostrados na Figura 6. As melhores médias '70-5', '70-8' e '70-10' tiveram seus intervalos

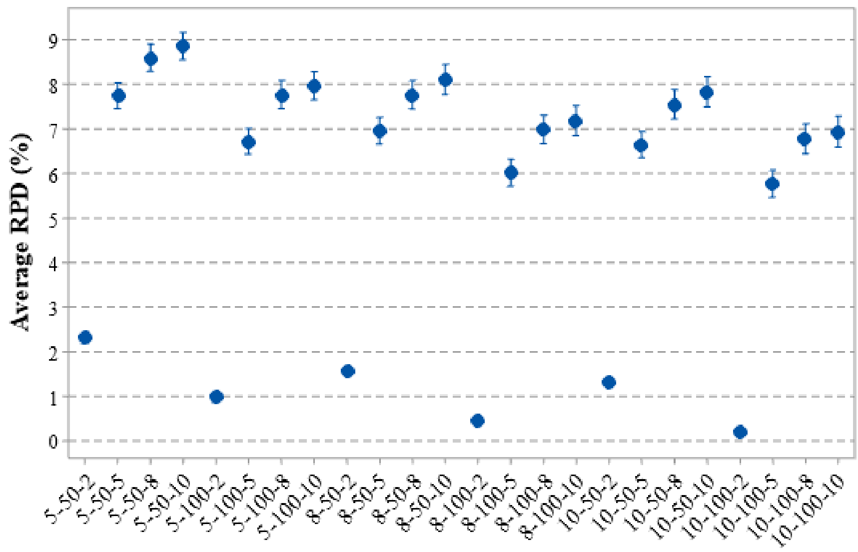


Fig. 5. Mean plot and confidence intervals at 95% confidence level for ILS_RVND_2 experiment.

sobrepostos, o que indica que são estatisticamente equivalentes. Portanto, foi escolhida a combinação '70-5' por executar em menor tempo do que as demais sobrepostas.

10.2 Resultados instâncias de pequeno porte

Nas instâncias pequenas, as heurísticas foram comparadas aos resultados o software CPLEX. Este foi definido com tempo limite máximo para execuções $t_{max} = 3600$. Para $N = 8$, tanto as heurísticas quanto o software atingiram a solução ótima. Dessa forma, as análises nesta subseção tratam as instâncias em que $N \in \{10, 15, 20\}$.

10.3 Resultados instâncias de grande porte

11 Conclusions

Acknowledgments. The authors thanks the financial support of FAPEMIG, CAPES and CNPq, Brazilian research agencies.

References

J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is np-hard," *Mathematics of operations research*, vol. 15, no. 3, pp. 483–495, 1990.

Table 3. Best RPD and Avg. RPD analyisis over small instances.

	ILS_RVND_1		ILS_RVND_2		GA_LS		CPLEX	
N x K	Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
10 x 3	0.00	0.01	0.00	0.04	0.00	0.00	0.45	0.45
10 x 4	0.00	0.00	0.02	0.04	0.00	0.00	0.36	0.36
10 x 5	0.00	0.00	0.00	0.05	0.00	0.00	0.38	0.38
10 x 6	0.00	0.00	0.01	0.05	0.00	0.00	0.35	0.35
15 x 3	0.00	0.00	0.70	1.96	0.00	0.05	5.98	5.98
15 x 4	0.00	0.00	0.21	0.77	0.00	0.03	5.07	5.06
15 x 5	0.00	0.00	0.41	0.89	0.03	0.04	3.64	3.63
15 x 6	0.00	0.00	0.27	0.60	0.00	0.02	3.75	3.74
20 x 3	0.00	0.00	3.76	5.79	0.07	0.22	21.28	21.21
20 x 4	0.00	0.01	2.68	4.45	0.09	0.30	21.28	21.22
20 x 5	0.00	0.01	1.14	2.21	0.00	0.11	18.53	18.53
20 x 6	0.03	0.00	1.05	1.81	0.01	0.13	15.63	15.51
Average:	0.00	0.00	0.86	1.55	0.02	0.08	8.06	8.03

Table 4. Best RPD and Avg. RPD analyisis over large instances.

	ILS_RVND_1		ILS_RVND_2		GA_LS	
N x K	Best	Avg.	Best	Avg.	Best	Avg.
50 x 5	0.41	0.28	12.02	13.23	0.21	0.87
50 x 8	0.00	0.00	10.90	12.68	0.93	1.80
50 x 10	0.00	0.00	10.11	11.51	1.15	2.06
50 x 12	0.00	0.00	9.05	10.41	1.40	2.12
80 x 5	0.71	0.57	14.52	14.61	0.89	1.34
80 x 8	0.05	0.00	16.53	17.30	1.76	2.72
80 x 10	0.00	0.00	16.13	16.93	2.19	2.82
80 x 12	0.00	0.00	14.56	15.73	2.11	2.88
100 x 5	0.61	0.22	14.63	14.80	0.91	1.41
100 x 8	0.00	0.00	17.12	18.09	2.03	2.70
100 x 10	0.01	0.00	16.86	18.01	2.24	3.09
100 x 12	0.00	0.00	16.64	17.64	2.37	3.02
Average:	0.15	0.09	14.09	15.08	1.52	2.24

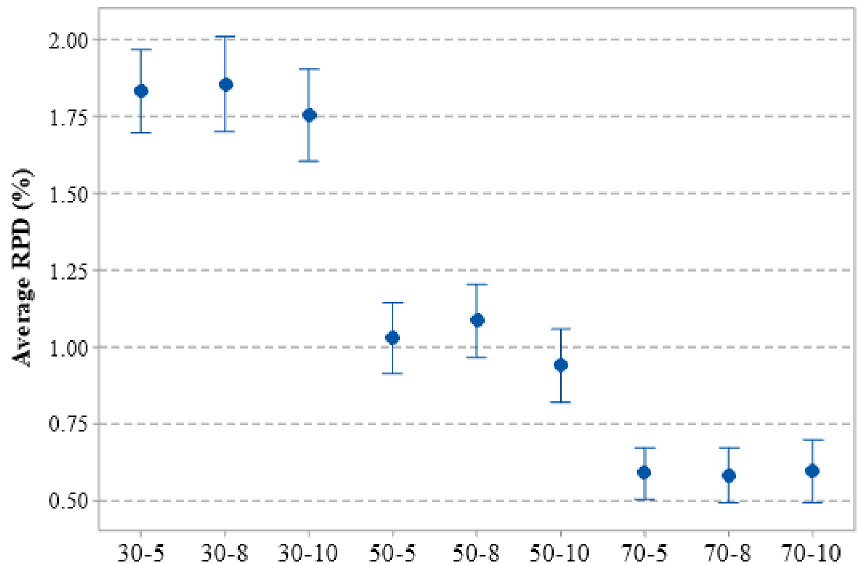


Fig. 6. Mean plot and confidence intervals at 95% confidence level for GA_LS experiment.

K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

M. Gendreau, G. Laporte, C. Musaraganyi, and É. D. Taillard, "A tabu search heuristic for the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, vol. 26, no. 12, pp. 1153–1173, 1999.

C. A. Ullrich, "Integrated machine scheduling and vehicle routing with time windows," *European Journal of Operational Research*, vol. 227, no. 1, pp. 152–165, 2013.

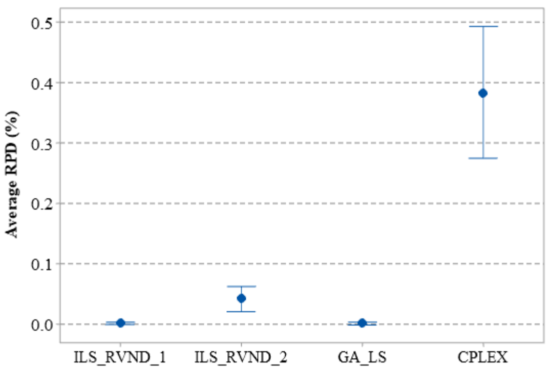
N. G. Hall and C. N. Potts, "Supply chain scheduling: Batching and delivery," *Operations Research*, vol. 51, no. 4, pp. 566–584, 2003.

G. Kirlik and C. Oguz, "A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine," *Computers & Operations Research*, vol. 39, no. 7, pp. 1506–1520, 2012.

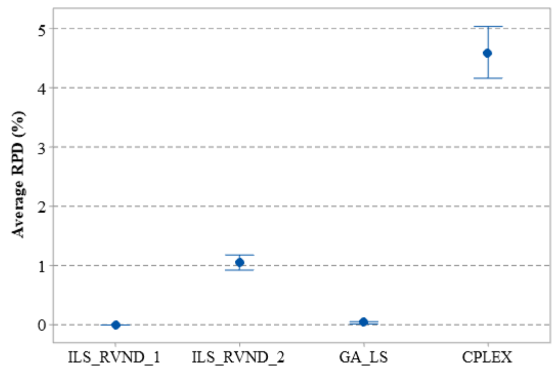
A. Condotta, S. Knust, D. Meier, and N. V. Shakhlevich, "Tabu search and lower bounds for a combined production–transportation problem," *Computers & operations research*, vol. 40, no. 3, pp. 886–900, 2013.

A. Hassanzadeh and M. Rasti-Barzoki, "Minimizing total resource consumption and total tardiness penalty in a resource allocation supply chain scheduling and vehicle routing problem," *Applied Soft Computing*, vol. 58, pp. 307–323, 2017.

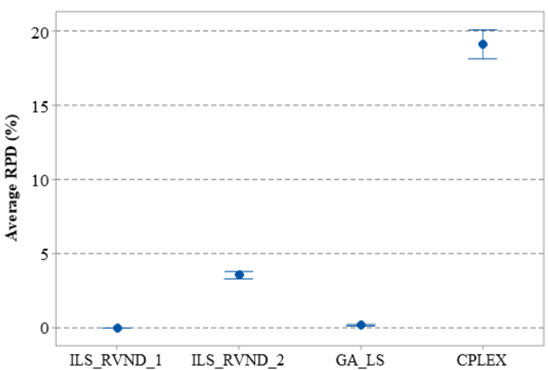
M. Tamannaei and M. Rasti-Barzoki, "Mathematical programming and solution approaches for minimizing tardiness and transportation costs in the supply chain scheduling problem," *Computers & Industrial Engineering*, vol. 127, pp. 643–656, 2019.



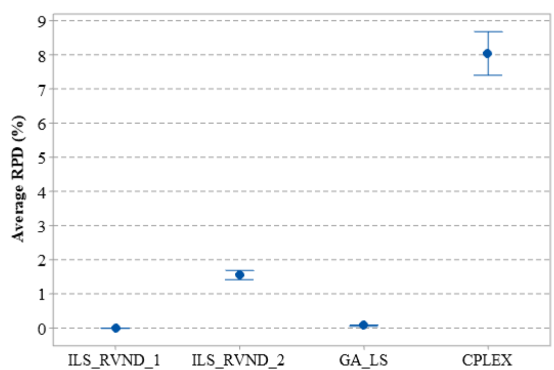
(a) Average RPD with $N = 10$



(b) Average RPD with $N = 15$

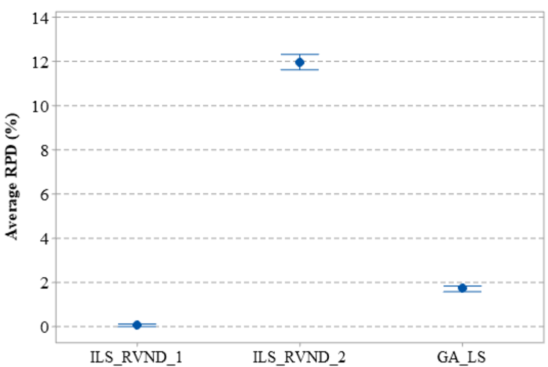


(c) Average RPD with $N = 20$

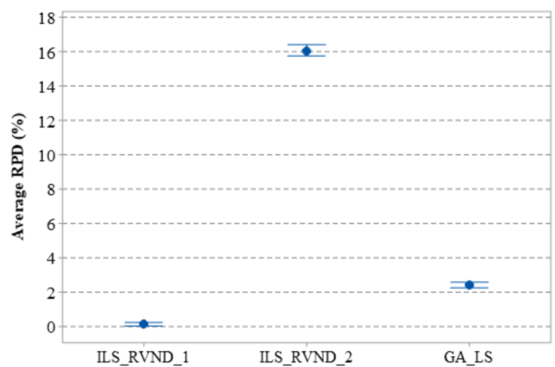


(d) General Average RPD

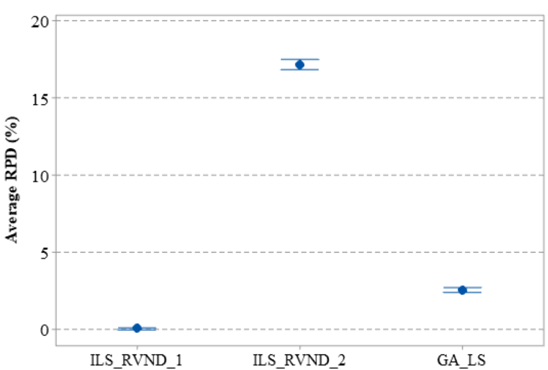
Fig. 7. Gráfico médio e intervalos de confiança com nível de confiança de 95% para instâncias de pequeno porte.



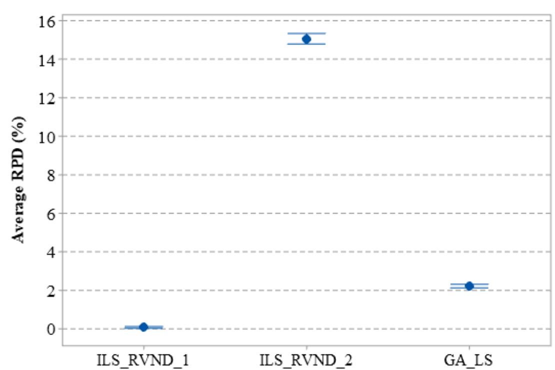
(a) Average RPD with $N = 50$



(b) Average RPD with $N = 80$



(c) Average RPD with $N = 100$



(d) General Average RPD

Fig. 8. Gráfico médio e intervalos de confiança com nível de confiança de 95% para instâncias de grande porte.

Z.-L. Chen, "Integrated production and outbound distribution scheduling: review and extensions," *Operations research*, vol. 58, no. 1, pp. 130–148, 2010.

N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

L. Molina-Sánchez and E. González-Neira, "Grasp to minimize total weighted tardiness in a permutation flow shop environment," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 161–176, 2016.

P. H. V. Penna, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the heterogeneous fleet vehicle routing problem," *Journal of Heuristics*, vol. 19, no. 2, pp. 201–232, 2013.