

# 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

Na literatura existem alguns trabalhos sobre o problema integrado de agendamento de produção e entrega.

Hall and Potts (2003) [2] 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. Em seu trabalho, o objetivo principal era minimizar os custos de agendamento e entrega, utilizando diversos objetivos clássicos de agendamento. A cooperação entre a manufatura e o fornecedor foi apresentada como fator importante de diminuição de custos no agendamento.

Ullrich (2013) [12] aborda um problema integrado que consiste na programação de um conjunto de tarefas em máquinas paralelas com tempos de espera dependentes da máquina. As tarefas processadas são distribuídas utilizando uma frota heterogênea de veículos. Este autor considera tempos de processamento, janelas

de tempo de entrega e tempos de serviço. Para minimizar o atraso total das tarefas, o autor apresenta um modelo de PLIM, duas abordagens clássicas de decomposição e um Algoritmo Genético.

Condotta, Knust, Meier, and Shakhlevich (2013) [3] 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.

Xiao, Konak (2015) [4] apresentaram o problema de Green Vehicle Routing and Scheduling (GVRSP) considerando condições dependentes do tempo de tráfego com múltiplas funções objetivo, como reduzir emissão total de CO<sub>2</sub>, distância total percorrida e tempo total de trajeto. Uma nova formulação matemática é proposta, e também um algoritmo de Simulated Annealing (SA) para resolver instâncias de grande porte. Resultados mostram que até 50% da emissão de CO<sub>2</sub> pode ser atingida, com reduções médias de 12% e 28% comparado à métodos tradicionais de roteamento.

Cheng, Leung and Li (2015) [5] estudaram o problema integrado de produção e entrega considerando o processamento em lotes em uma máquina, desde que respeite sua capacidade. Em seu trabalho, tamanhos das tarefas e tempos de processamento são arbitrários, e veículos com capacidades idênticas. Foi gerado um limite inferior (lower bound) para o custo total ideal, e implementado um algoritmo de Colônia de Formigas, obtendo bons resultados e executando em tempo curto para instâncias de grande porte.

Há pouco tempo, Hassanzadeh and RastiBarzoki (2017) [6] estudaram a otimização na cadeia de produção considerando o problema de roteamento de veículos com penalidades de atraso e consumo total de recursos.

Billaut, Croce and Ta (2017) [7] abordaram o problema integrado de agendamento de produção em entrega em um ambiente de produção Flow Shop com apenas um veículo disponível. Foi minimizado o atraso total das entregas, sendo implementados dois algoritmos de Tabu Search e uma metaheurística (matheuristic). Experimentos foram realizados em conjuntos de dados aleatórios.

Wang, Yao, Sheng and Yang (2019) [8] estudaram o problema integrado de produção e distribuição de tarefas para uma máquina linear e múltiplos veículos. Neste ambiente, é possível trocar uma máquina entre duas produções adjacentes. O objetivo de minimizar a emissão total de carbono foi acrescentado à minimizar os custos totais. Foram desenvolvidos um modelo MILP para a solução e em seguida um algoritmo híbrido de Busca Tabu.

Ghannadpour and Zarrabi (2019) [9] abordaram o problema multi-objetivo de frota heterogênea integrada a produção, com minimização de energia de acordo com novos requisitos sustentáveis. São apresentados dois casos de minimização, são tratados o número de veículos alugados e a satisfação dos clientes, acrescentado à minimização de energia (primeiro caso) e minimização de distância percorrida(segundo caso). É apresentada uma nova formulação matemática para o problema de roteamento de veículos com janelas de tempo (VRPTW) e desenvolvido uma nova solução baseada em um algoritmo evolucionário.

Liu, Wenli and Kunpeng Li, and Zou (2020) [10] estudaram o problema integrado de produção e entrega para uma máquina linear, com objetivo de minimizar a soma dos tempos de entrega. Neste trabalho, os veículos são limitados e possuem mesma capacidade. É proposto um algoritmo de Variable Neighborhood Search (VNS) e complementado com um algoritmo de Busca Tabu para intensificação. Os resultados são comparados com o Solver CPLEX e dois algoritmos de heurísticas na literatura, obtendo-se bons resultados.

Tamannaei and Morteza (2019) [11] 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 início  $S_k$ . 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:

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

## 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) [11], sendo o primeiro desenvolvido na versão do problema em que veículos possuem mesma capacidades ( $Q$ ) e custos ( $F$ ). O modelo é descrito a seguir:

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

st.

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

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

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

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

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

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

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

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

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

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

$$T_i \geq D_i - d_i; \quad \forall i = 1 \dots N; \quad (12)$$

$$C_0 = 0; \quad (13)$$

$$D_0 = 0; \quad (14)$$

$$T_0 = 0; \quad (15)$$

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

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

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

$$S_k \geq 0; \quad \forall k = 0 \dots K; \quad (19)$$

$$T_i \geq 0; \quad \forall i = 1 \dots N; \quad (20)$$

$$D_i \geq 0; \quad \forall i = 1 \dots N; \quad (21)$$

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 greater 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.

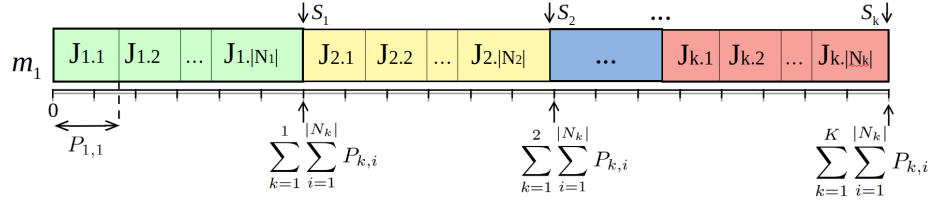
Constraints (16), (17) and (18) are to define these variables as binary variables. Constraints (19), (20) and (21) ensure that these variables won't assume negative values.

## 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 início 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:

$$R_k = O \rightarrow J_1 \rightarrow J_2 \rightarrow \dots \rightarrow J_{|N_k|} \rightarrow O \quad (22)$$

Neste contexto,  $J_i$  são as tarefas entregues no trajeto e  $\forall i = 1, \dots, |N_k| \in N$  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.

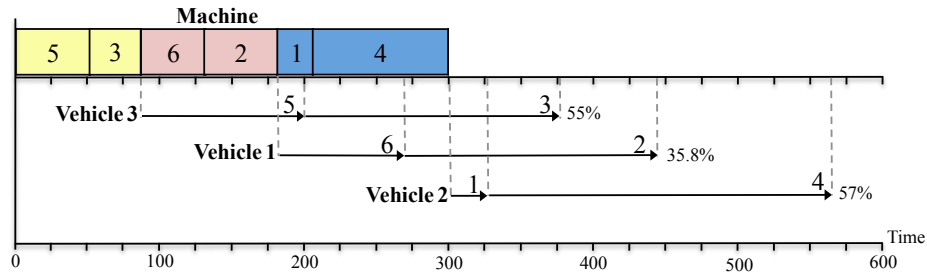
Os tempos de destino  $D$ , que indicam o momento em que o pedido chega aos clientes são: 328, 445, 378, 565, 200 e 270.

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.



**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.

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 distribuídas 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:

$$P_i \in rand[1, \rho], \quad with \quad \rho = 100 \quad (1)$$

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

$$s_i \in rand[1, P_i] \quad (2)$$

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  $\bar{z}_i$ :

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

Os parâmetros  $\pi_1$  e  $\pi_2$  influenciam na dificuldade da instância pois seus valores podem encurtar ou prolongar as janelas de tempo, resultando em prazos de entrega menores ou maiores:

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

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

$$w_i \in rand[1.0, 5.0] \quad (5)$$

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:

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

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

$$F_k = N * Q_k \quad (7)$$

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:

$$x_0, y_0 \in rand[400, 700] \quad (8)$$

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:

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

Por fim, são indicados os parâmetros utilizados nas equações acima ( $\delta, \mu, N$  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  $\mu, N$  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  $\delta$  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 ( $\delta, \mu, N$  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) [12] 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) [13], 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) [14]. 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) [15].

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 Vizinhaça

O processo de busca local visa aprimorar uma solução atual ao considerar soluções similares pelas operações de vizinhaça 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:

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

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

## 7.3 Vizinhaças 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 Vizinhaças Inter-Rota

As operações de vizinhaça 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) [13], 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 melhora 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 distribuído 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.

A segunda versão, RVND-Custom, é baseada na técnica de aprimoramento interno (Penna and Ochi, 2013) [16], 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 distribuído e armazenado em *InterRoute* (Passo 2). Todos as vizinhanças são visitadas na ordem sorteada (Passos 3-22), sendo

---

**Algorithm 1:** RVND( $S$ )

---

**Input** : Solution  $S$   
**Output** : Better or equal  $S^*$  solution

```
1  $S^* \leftarrow S$ ;  
2  $NeighborSet \leftarrow \text{Random Order}(Neighborhoods)$ ;  
3 for  $i \leftarrow 1$  in  $NeighborSet$  do  
4    $currNeighborhood \leftarrow NeighborSet[i]$ ; // Select Neighborhood  
5    $S \leftarrow currNeighborhood(S)$ ; // Apply Local Search  
6   if  $f(S) \leq f(S^*)$  then  
7      $S^* \leftarrow S$ ;  
8      $NeighborSet \leftarrow \text{Random Order}(Neighborhoods)$ ;  
9      $j \leftarrow 1$ ; // Restart Local Search  
10  end  
11 end  
Output :  $S^*$ 
```

---

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) [14]. 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  $\alpha$  de perturbações, sendo  $\alpha$  estatisticamente definido na Seção ‘Calibração de Parâmetros’.

### 8.1 ILS RVND

O esquema geral da metaheurística ILS proposta é apresentado em Algoritmo 3. Existem três parâmetros:  $\alpha$  (número de perturbações),  $MaxIter$  (quantidade de reinícios de solução) e  $MaxIter_{ILS}$  (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  $S$

---

**Algorithm 2:** RVND-Custom( $S$ )

---

**Input** : Solution  $S$   
**Output** : Better or equal  $S^*$  solution

```
1  $S^* \leftarrow S$ ;  
2  $InterRoute \leftarrow$  Random Order(Inter-Route Neighborhoods);  
3 for  $i \leftarrow 1$  in  $InterRoute$  do  
4    $currInter \leftarrow InterRoute[i]$ ; // Get Inter-Route Neighborhood  
5    $S' \leftarrow currInter(S)$ ; // Apply Inter Local Search  
6   if  $f(S') \leq f(S)$  then  
7      $IntraRoute \leftarrow$  Intra-Route Neighborhoods;  
8     for  $j \leftarrow 1$  in  $IntraRoute$  do  
9        $currIntra \leftarrow IntraRoute[j]$ ; // Apply Intra Local Search  
10       $S'' \leftarrow currIntra(S')$ ;  
11      if  $f(S'') \leq f(S')$  then  
12         $S' \leftarrow S''$ ;  
13         $j \leftarrow 1$ ; // Restart Intra-Route Social Search  
14      end  
15    end  
16     $i \leftarrow 1$ ;  
17     $InterRoute \leftarrow$  Random Order(Inter-Route Neighborhoods);  
18  end  
19  if  $f(S') \leq f(S^*)$  then  
20     $S^* \leftarrow S'$ ;  
21  end  
22 end  
Output :  $S^*$ 
```

---

(Passo 4). Após sua possível melhora, o processo de Perturbação é inicializado (Passos 5-12), a solução atual  $S'$  é perturbada  $\alpha$  vezes (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  $S^* \leftarrow \text{Initialize}(S^*)$ ;
2 for  $i \leftarrow 1$  to  $MaxIter$  do
3    $S \leftarrow \text{GenerateInitialSolution}()$ ;
4    $S' \leftarrow \text{RVND}(S)$ ;
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     $S' \leftarrow \text{Perturb}(S', \alpha)$ ;
11     $S' \leftarrow \text{RVND}(S')$ ; // 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  $S^*$ .

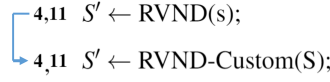
## 8.2 ILS-RVND com Aprimoramento de Rota Interna

Penna and Ochi, 2013 [16] 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.

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.





**Fig. 3.** Mudança entre acionamentos de RVND.

## 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) [14]. 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).
- 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

---

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

---

**Input** : Population size and max. iteration number  
**Output** : Best solution  $P^*$  in  $MaxIter$  populations

```
1  $P \leftarrow$  Generate random population( $PopSize$ );
2  $P \leftarrow$  RVND( $P$ ); // Apply local search to new population P
3  $P^* \leftarrow$  Best( $P$ ); // Save best solution of P
4 for  $i \leftarrow 1$  to  $MaxIter$  do
5    $P' \leftarrow \emptyset$ ;
6   for  $j \leftarrow 1$  to  $(PopSize - 1)$  do
7      $Parent1 \leftarrow$  Binary Tournament(random( $P$ ), random( $P$ ));
8      $Parent2 \leftarrow$  Binary Tournament(random( $P$ ), random( $P$ ));
9      $Offspring1 \leftarrow$  Crossover( $first, second$ );
10     $Offspring2 \leftarrow$  Crossover( $first, second$ );
11    if Feasible( $Offspring1$ ) and Feasible( $Offspring2$ ) then
12       $P' \leftarrow P' +$  BinaryTournament( $Offspring1, Offspring2$ );
13    else
14      if Feasible( $Offspring1$ ) or Feasible( $Offspring2$ ) then
15         $P' \leftarrow P' +$  Feasible from( $Offspring1, Offspring2$ );
16      else
17        if !Feasible( $Offspring1$ ) and !Feasible( $Offspring2$ ) then
18           $P' \leftarrow P' +$  Mutation(random( $Parent1, Parent2$ ));
19        end
20      end
21    end
22  end
23   $P' \leftarrow P' + P^*$ ; // Include best solution to P'
24   $P' \leftarrow$  Fast Local Search( $P'$ );
25  if  $f(P') \leq f(\text{Best}(P'))$  then
26     $P' \leftarrow P^*$ ;
27  end
28   $P \leftarrow P'$ ;
29 end
```

**Output** :  $P^*$

---

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  $NeighborhoodSet \leftarrow Inter\text{-}Route \text{ and } Intra\text{-}Route \text{ Neighborhoods}$ ;
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 atribuído 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 16.04 64 bits.

A métrica de comparação utilizada é o desvio relativo percentual (RPD), sendo utilizada o melhor resultado 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:

$$RPD = \frac{f_{method} - f_{best}}{f_{best}} * 100\% \quad (14)$$

Também são utilizadas as métricas Average RPD(%) e Best RPD(%), onde a primeira utiliza o  $f_{method}$  como sendo a média dos resultados das execuções e a segunda como o melhor resultado dentre as execuções. A performance das heurísticas aplicadas foram testadas em todas as 1740 instâncias.

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 Parameters Calibration

Nesta Seção, testes preliminares são apresentados para definir os parâmetros de execução dos três algoritmos.

O primeiro algoritmo, ILS\_RVND\_1, utiliza três parâmetros: o número de perturbações  $\alpha$ , 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, sendo cada uma delas executada 10 vezes para as 1740 instâncias.

Os resultados foram analisados por meio do teste não-paramétrico de Kruskal-Wallis utilizando o Average RPD(%) como variável de resposta. Seguindo a Equação 14, a variável  $f_{method}$  foi definida como sendo o resultado da combinação para a instância e  $f_{best}$  como o melhor resultado obtido dentre todas as combinações. No teste de Kruskal-Wallis duas hipóteses são testadas: a hipótese nula afirma que a performance de todas as 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.

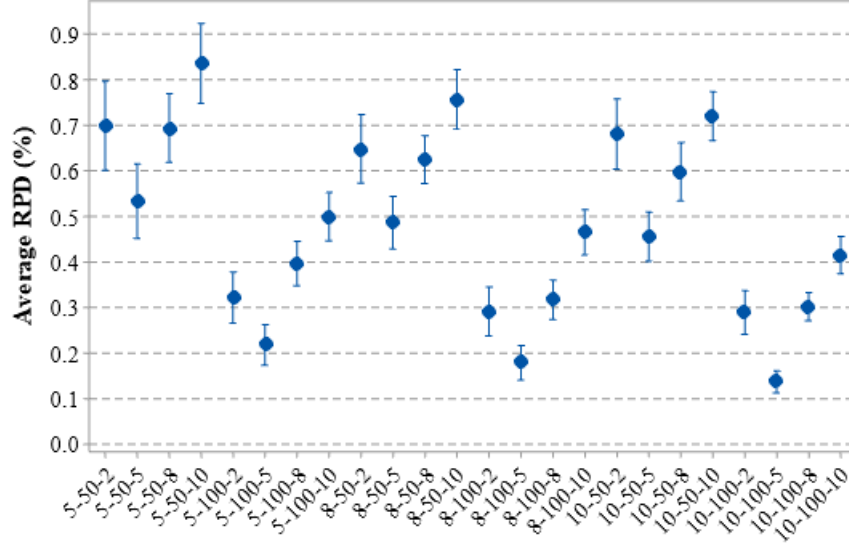
Intervalos que se sobrepõem sugerem que pode não haver diferença entre as implementações. Os intervalos '8-100-5' e '10-100-5' se sobrepuseram e apresentaram com menor média. Pelo teste de Kruskal-Wallis, o p-valor=0.00  $\leq$  0.05 obtido indica que a hipótese nula deve ser rejeitada, concluindo assim que pelo menos uma implementação tem performance significativamente diferente (hipótese alternativa). A combinação '10-100-5' foi escolhida por apresentar o melhor intervalo.

De forma análoga, para o segundo algoritmo ILS\_RVND\_2, foram utilizadas as 24 combinações de parâmetros acima e em seguida executadas 10 vezes em cada instância. As combinações que possuíam menor o número de perturbações  $\alpha$  obtiveram destaque (Figura 5).

As duas menores médias '8-100-2' e '10-100-2' não se sobrepuseram nos intervalos de confiança, indicando que há diferenças entre as implementações. Ademais, com o p-valor=0.00  $\leq$  0.05, a hipótese alternativa é novamente válida e há pelo menos uma implementação com performance significativamente diferente. Isto posto, foi definida a combinação '10-100-2' para o algoritmo.

O terceiro algoritmo, GA\_LS, utiliza dois parâmetros:  $PopSize$  e  $MaxIter$ , o tamanho da população e o número máximo de iterações, respectivamente. Os conjuntos de valores destes parâmetros foram definidos em  $PopSize \in \{30, 50, 70\}$  e  $MaxIter \in \{5, 8, 10\}$ . São geradas 9 combinações de parâmetros e foram executadas em todas as instâncias.

Novamente, pelo teste de Kruskal-Wallis, o p-valor=0.00  $\leq$  0.05 obtido indica a validade da hipótese alternativa. Os intervalos de confiança são mostrados



**Fig. 4.** Mean plot and confidence intervals at 95% confidence level for ILS.RVND.1 experiment.

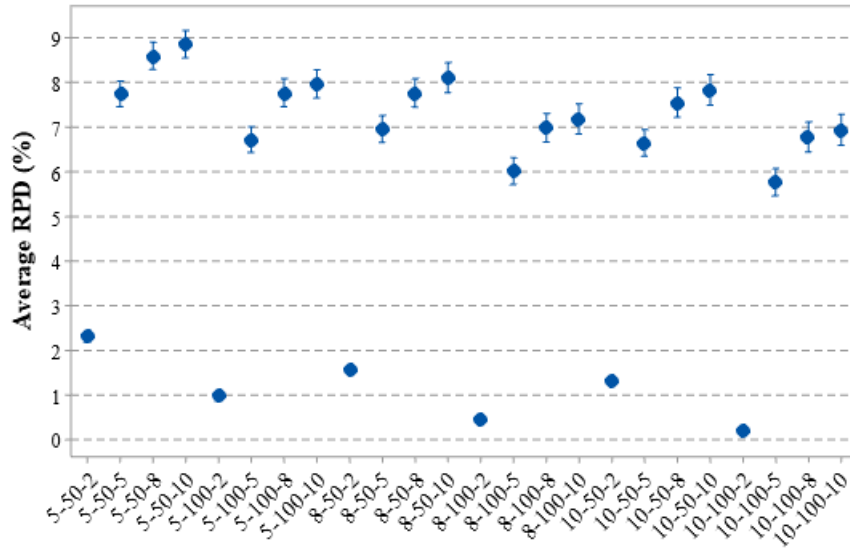
na Figura 6. As melhores médias '70-5', '70-8' e '70-10' tiveram seus intervalos sobrepostos, o que indica que são estatisticamente equivalentes. Portanto, foi escolhida a combinação '70-5' por executar em menor tempo do que as demais ao possuir um número menor de iterações.

## 10.2 Small-scale Instances Results

Em instâncias pequenas, os resultados das heurísticas foram comparados entre si e aos resultados do software CPLEX. Para o software, foi definido o tempo limite de execução de  $t_{max} = 3600s$ . Para  $N = 8$ , tanto as heurísticas quanto o software atingiram a solução ótima. Dessa forma, as análises nesta Seção tratam as instâncias em que  $N \in \{10, 15, 20\}$ . Todas as três heurísticas foram executadas cinco vezes para cada uma das 1200 instâncias pequenas, e o software CPLEX uma única vez.

A performance de cada algoritmo é medida pelo Average RPD(%) seguindo a Equação 14, onde  $f_{best}$  é o melhor resultado encontrado pelos quatro algoritmos, e  $f_{method}$  como sendo a média das cinco execuções da heurística para cada instâncias. Como o software CPLEX foi executado uma vez,  $f_{method}$  é o resultado desta execução.

Os resultados para as quatro implementações foram analisados pelo teste de Kruskal-Wallis. O p-valor= $0.00 \leq 0.05$  obtido indica que ao menos uma das implementações é significativamente diferente. Na Figura 7 são plotados os



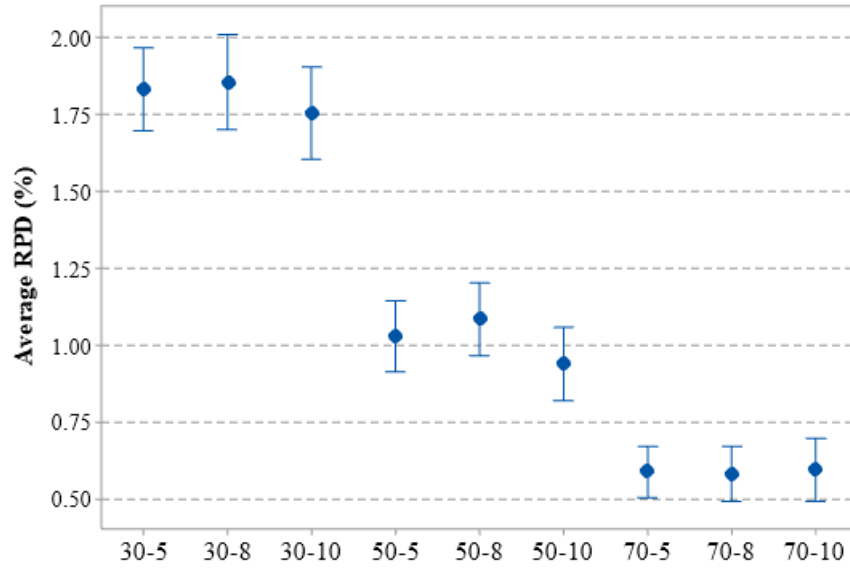
**Fig. 5.** Mean plot and confidence intervals at 95% confidence level for ILS\_RVND.2 experiment.

valores de Average RPD(%) das instâncias agrupadas pelo número de tarefas, e por último um gráfico geral de todas instâncias.

É perceptível o distanciamento dos resultados do software CPLEX a medida em que o número de tarefas aumenta. Os algoritmos ILS\_RVND.1 e GA\_LS obtiveram seus intervalos sobrepostos para  $N = 10$  e  $N = 15$ , indicando que suas implementações são estatisticamente equivalentes para estes grupos. No gráfico geral, o algoritmo ILS\_RVND.1 obteve a melhor média, seguido pelas heurísticas GA\_LS, ILS\_RVND.2 e então CPLEX.

Na Tabela 3, as instâncias são agrupadas pelo número de tarefas  $N$  e pelo número de veículos  $K$ , e seus resultados avaliados pelo Best RPD(%) e Average RPD(%). Nesta tabela, é possível observar a qualidade das soluções ao aumentar o número de clientes e veículos. Os resultados retratam o cenário apresentado acima, onde a heurística ILS\_RVND.1 possui a menor média geral. Este algoritmo disputa os melhores resultados com o Algoritmo Genético GA\_LS em 66,67% dos grupos, e possui a média para o Best RPD(%) quando comparada à heurística GA\_LS para  $N = 20$  e  $K = 6$ .

Um fator importante a se considerar ao julgar a qualidade de soluções dos algoritmos é o tempo de execução. Neste trabalho, o tempo foi medido em segundos e suas médias apresentadas por grupos na Tabela 4. Nela, o software CPLEX é executado com tempo médio  $t=3600(s)$ , indicando que a solução ótima não foi atingida com certeza no tempo limite. Para as três heurísticas, o pior tempo



**Fig. 6.** Mean plot and confidence intervals at 95% confidence level for GA\_LS experiment.

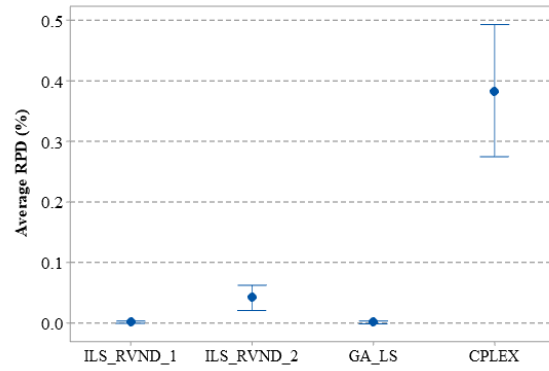
médio foi alcançado pela heurística populacional (GA\_LS). Dentre as heurísticas de solução única, ILS\_RVND\_1 obteve o melhor tempo médio de execução.

### 10.3 Large-scale Instances Results

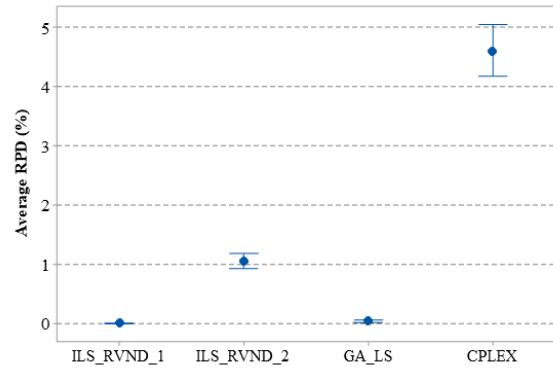
Em instâncias grandes, os resultados para as heurísticas foram comparados entre si. As três heurísticas foram executadas cinco vezes em cada uma das 540 instâncias de grande porte. A qualidade das soluções encontradas é também avaliada pelo Average RPD(%) pela Equação 14 e plotadas na Figura 10.3 com intervalos de à 95% de confiança. Neste grupo de instâncias, é possível observar um distanciamento considerável do algoritmo ILS\_RVND\_2 para os demais algoritmos.

Pelo Teste de Kruskal-Wallis, o  $p\text{-valor}=0.00 \leq 0.05$  obtido sustenta a hipótese alternativa de que pelo menos uma implementação é significativamente diferente das demais. Além disso, não existem intervalos sobrepostos nos gráficos para  $N \in \{50, 80, 100\}$ , o que indica que todas os resultados são realmente estatisticamente diferentes.

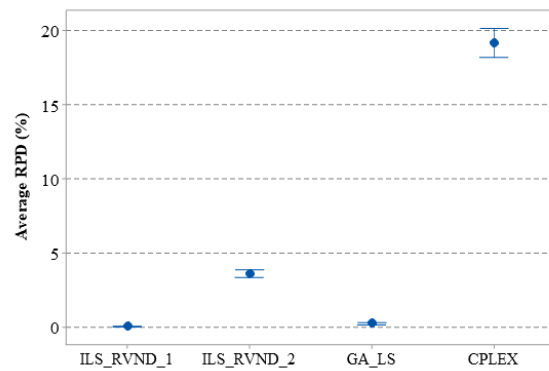
Os algoritmos se comportam de forma semelhante à medida em que o número de clientes  $N$  aumenta, e no gráfico geral, ao distanciar o intervalo algoritmo ILS\_RVND\_1 do eixo x, indica que há instâncias em que os demais algoritmos obtiveram melhores resultados.



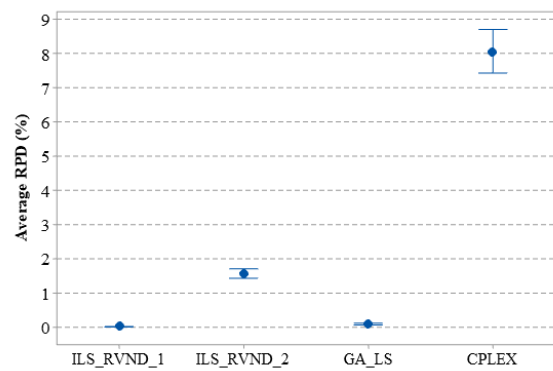
(a) Average RPD with  $N = 10$



(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.

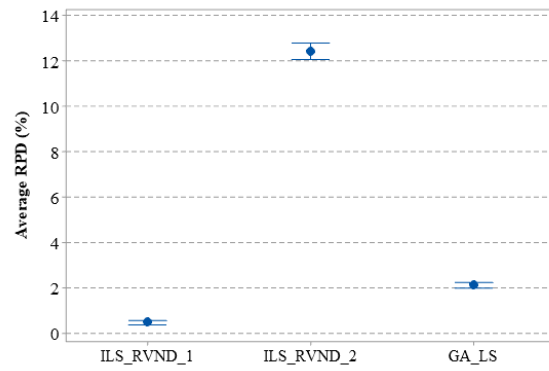


**Table 3.** Best RPD and Avg. RPD analysis over small instances.

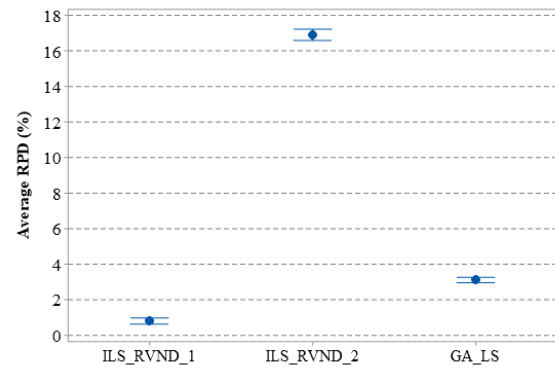
N x K	ILS_RVND_1		ILS_RVND_2		GA_LS		CPLEX
	Best	Avg.	Best	Avg.	Best	Avg.	Best
10 x 3	0.00	0.01	0.00	0.04	0.00	0.00	0.45
10 x 4	0.00	0.00	0.02	0.04	0.00	0.00	0.36
10 x 5	0.00	0.00	0.00	0.05	0.00	0.00	0.38
10 x 6	0.00	0.00	0.01	0.05	0.00	0.00	0.35
15 x 3	0.00	0.00	0.70	1.96	0.00	0.05	5.98
15 x 4	0.00	0.01	0.21	0.77	0.00	0.04	5.07
15 x 5	0.00	0.02	0.41	0.90	0.03	0.06	3.64
15 x 6	0.00	0.00	0.27	0.60	0.00	0.03	3.75
20 x 3	0.00	0.06	3.76	5.86	0.07	0.29	21.28
20 x 4	0.00	0.05	2.68	4.50	0.09	0.35	21.28
20 x 5	0.00	0.01	1.14	2.21	0.00	0.12	18.53
20 x 6	0.03	0.11	1.05	1.92	0.01	0.24	15.63
Average:	0.00	0.02	0.86	1.58	0.02	0.10	8.06

**Table 4.** Runtime Average Analysis over small instances.

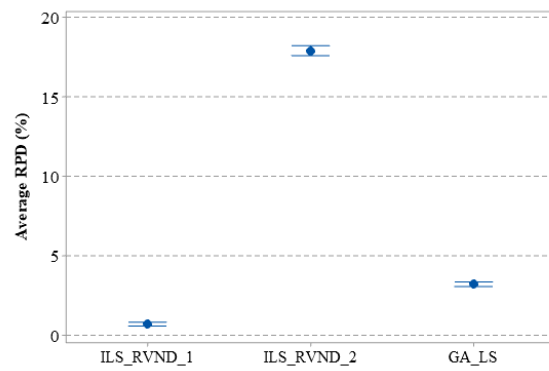
N x K	ILS_RVND_1	ILS_RVND_2	GA_LS	CPLEX
	Avg.	Avg.	Avg.	Avg.
10 x 3	0.11	0.16	0.17	3575.10
10 x 4	0.14	0.25	0.22	3620.62
10 x 5	0.17	0.37	0.27	3629.82
10 x 6	0.20	0.45	0.33	3634.85
15 x 3	0.33	0.30	0.53	3614.83
15 x 4	0.39	0.46	0.62	3632.21
15 x 5	0.47	0.71	0.74	3647.36
15 x 6	0.54	0.91	0.86	3649.92
20 x 3	0.79	0.51	1.25	3612.73
20 x 4	0.83	0.76	1.34	3615.70
20 x 5	0.99	1.11	1.61	3618.98
20 x 6	1.09	1.52	1.74	3621.38
Average:	0.51	0.63	0.81	3622.79



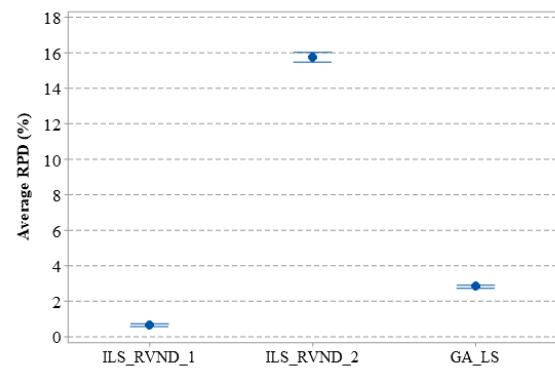
(a) Average RPD with  $N = 50$



(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.

Na Tabela 5, ao analisar os resultados pelos conjuntos de instâncias, é possível observar que a primeira heurística ILS\_RVND\_1 obteve destaque em cima das demais, seguido do Algoritmo Genético GA\_LS e da segunda heurística ILS\_RVND\_2. Apenas em um conjunto de instâncias, '50-5', o primeiro obteve a média para o Best RPD(%) inferior a média de melhor resultado do Algoritmo Genético.

**Table 5.** Best RPD and Avg. RPD analysis over large instances.

N x K	ILS_RVND_1		ILS_RVND_2		GA_LS	
	Best	Avg.	Best	Avg.	Best	Avg.
50 x 5	0.41	0.89	12.02	13.91	0.21	1.48
50 x 8	0.00	0.41	10.90	13.14	0.93	2.22
50 x 10	0.00	0.31	10.11	11.86	1.15	2.38
50 x 12	0.00	0.31	9.05	10.75	1.40	2.43
80 x 5	0.71	1.63	14.52	15.80	0.89	2.40
80 x 8	0.05	0.65	16.53	18.05	1.76	3.38
80 x 10	0.00	0.59	16.13	17.63	2.19	3.43
80 x 12	0.00	0.40	14.56	16.19	2.11	3.29
100 x 5	0.61	1.24	14.63	15.95	0.91	2.44
100 x 8	0.00	0.69	17.12	18.90	2.03	3.40
100 x 10	0.01	0.46	16.86	18.56	2.24	3.57
100 x 12	0.00	0.45	16.64	18.17	2.37	3.48
Average:	0.15	0.67	14.09	15.74	1.52	2.83

Ao analisar o tempo de execução das três implementações em instâncias grandes, na Tabela 6 é possível observar que em todos os grupos, o algoritmo ILS\_RVND\_2 foi executou com média mais rápida, seguido do algoritmo ILS\_RVND\_1 e então do Algoritmo Genético GA\_LS. Por se tratar de uma heurística populacional, o GA\_LS consome mais tempo para gerar e avaliar toda a população repetidas vezes.

Para a heurística ILS\_RVND\_2, a técnica de Aprimoramento Interno de Rota não obteve sucesso para instâncias grandes, e isso se dá pelo aprimoramento de vizinhanças externas às rotas não obter um número favorável de melhoras a assim acionar o aprimoramento interno. Isso não acontece na primeira heurística ILS\_RVND\_1, que aciona todas as vizinhanças internas e externas em ordem aleatória (RVND). Dessa forma, o tempo de médio de execução para a heurística ILS\_RVND\_1 é significativamente superior a da outra (ILS\_RVND\_2).

**Table 6.** Runtime Average Analysis over large instances.

	ILS_RVND_1	ILS_RVND_2	GA_LS
N x K	Avg.	Avg.	Avg.
50 x 5	17.57	8.48	35.41
50 x 8	16.93	11.90	36.60
50 x 10	16.32	14.44	36.55
50 x 12	16.46	17.22	38.87
80 x 5	68.03	28.37	180.42
80 x 8	58.88	35.36	162.78
80 x 10	55.00	40.32	154.89
80 x 12	53.02	46.44	160.59
100 x 5	136.08	51.34	415.69
100 x 8	114.50	61.28	357.94
100 x 10	103.80	68.83	340.52
100 x 12	97.80	77.04	343.74
Average:	62.87	38.42	188.67

## 11 Conclusions

Neste trabalho foram realizados estudos para solucionar o problema integrado de agendamento de produção e entrega, com frota heterogênea de veículos e tarefas de diferentes capacidades.

Por se tratar de um problema NP-Difícil, métodos convencionais são ineficientes para resolver o problema. Pelo conhecimento dos autores, até o atual momento não é abordada na literatura esta versão para o problema integrado.

Foram propostos um modelo MILP, duas heurísticas Iterated Local Search com RVND e um Algoritmo Genético para o problema. As instâncias para esta nova abordagem foram geradas aqui, totalizando 1740 instâncias de pequeno e grande porte. A quantidade máxima de clientes considerados foi de 100 clientes.

Em instâncias pequenas, as heurísticas obtiveram melhores soluções do que o software CPLEX em um tempo muito inferior. A implementação básica de Iterated Local Search com RVND (ILS\_RVND\_1), no geral, se mostrou a mais eficiente quando comparadas com as demais e ao software CPLEX, em qualidade de solução e tempo de execução.

Para trabalhos futuros, o objetivo é tratar esse problema em um ambiente de máquinas paralelas e tempo de preparação das máquinas.

**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.
- N. G. Hall and C. N. Potts, "Supply chain scheduling: Batching and delivery," *Operations Research*, vol. 51, no. 4, pp. 566–584, 2003.
- 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.
- Y. Xiao and A. Konak, "A simulating annealing algorithm to solve the green vehicle routing & scheduling problem with hierarchical objectives and weighted tardiness," *Applied Soft Computing*, vol. 34, pp. 372–388, 2015.
- B.-Y. Cheng, J. Y.-T. Leung, and K. Li, "Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method," *Computers & Industrial Engineering*, vol. 83, pp. 217–225, 2015.
- 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.
- J.-C. Billaut, F. Della Croce, and Q. C. Ta, "Tabu search and matheuristic algorithms for solving an integrated flow shop and vehicle routing problem," in *12th Metaheuristics International Conference (MIC)*, 2017.
- J. Wang, S. Yao, J. Sheng, and H. Yang, "Minimizing total carbon emissions in an integrated machine scheduling and vehicle routing problem," *Journal of Cleaner Production*, vol. 229, pp. 1004–1017, 2019.
- S. F. Ghannadpour and A. Zarrabi, "Multi-objective heterogeneous vehicle routing and scheduling problem with energy minimizing," *Swarm and evolutionary computation*, vol. 44, pp. 728–747, 2019.
- L. Liu, W. Li, K. Li, and X. Zou, "A coordinated production and transportation scheduling problem with minimum sum of order delivery times," *Journal of Heuristics*, vol. 26, no. 1, pp. 33–58, 2020.
- M. Tamannaie 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.
- 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. 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.