Effective Heuristics for Integrated Job Scheduling and Heteregoneous Fleet Vehicle Routing Problem

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. This paper address an integrated job scheduling and vehicle routing problem, where jobs contain different sizes, penalties and due dates. To deliver them, there are vehicles from a limited heterogeneous fleet (HFVRP). The minimization criteria are total weighted penalty, transport costs and total distance traveled. The given problem is classified as NP-Hard, demanding heuristics approaches to obtain near-optimal solutions in reasonably computational time. There are proposed two versions of Iterated Local Search (ILS), one Genetic Algorithm (GA) and a Mixed-Integer Linear Programming (MILP) model to solve it. In addition, were also generated small and big instances (with at most one hundred customers) and the results compared to CPLEX Solver. Statistical Analysis shows that heuristics approaches obtained better quality solutions in much less time than the solver.
Keywords: Single machine scheduling, vehicle routing, heterogeneous fleet, time windows, linear programming, meta-heuristics.

1 Introduction

In order to improve competition in the global market, industries have been focusing on optimizing their supply chains. A supply chain is the set of tasks/activities involving production, storage and products or services transportation. These activities include buying raw material, stock control, production and delivery products to final customers according to due dates.

All activities must be well-planned and optimized to generate and delivery high quality products and obtain good results. The supply chains primordial objectives are costs reduction, follow pre established dates and customer satisfaction.

This work's objective is to propose a solution approach for the integrated job scheduling and vehicle routing problem. In this problem, a set of jobs with different sizes must be processed in a machine (available in a factory) and delivery to respective customers by a heterogeneous vehicle fleet, considering due dates.

The main objective is to minimize total penalty weight and delivery costs (considering different vehicle use costs and routes time/costs). Separately, the

job scheduling in a linear machine minimizing total tardiness is proved as NP-Hard (Du and Leung, 1990) [1]. Thus, this approach is also NP-Hard.

Job scheduling problems and vehicle routing problems are well-studied combinatorial optimization problems in literature, and are usually studied separately.

To the best of authors' knowledge here, the integrated problem of job scheduling in a linear machine and vehicle routing with vehicles having different capacities and variable costs, jobs with heterogeneous sizes, and focusing on minimize total penalty weight and delivery costs, is not addressed and discussed yet in literature. The main contributions of this article consist of:

- Two Iterated Local Search (ILS) heuristic approaches and one Genetic Algorithm (GA)
- A mixed integer linear programming for solving the integrated problem.

This article is organized as follows: in Section 2 are presented previous studies in literature. Following, in Sections 3 and 4 the problem description and the MILP model are shown. Solution representation is described in Section 5 added to a hypothetical case in Section 5.1.

In Section 6 are described the instances' generation. In sequence, RVND local search methods and neighborhood operators are presented in Section 7. The initial solutions criteria are shown in Section 8.

Both Iterated Local Search versions are described in 9, and the Genetic Algorithm in Section 10. Following, computational experiments and parameters calibration are reported in Section 11. Results for small and large instances are presented in Subsections 11.2 and 11.3, respectively.

Finally, the studied problem is concluded in Section 12.

2 Literature Review

In literature there are some previous works about the integrated job schedulling and vehicle routing problem.

Hall and Potts (2003) [2] approached the integrated job scheduling and batch delivery problem, with different customers and manufacturers, and so they defined the Supply Chain Scheduling area. In their work, the main object was minimizing scheduling and delivery's cost, by using classic scheduling objectives. Cooperation between manufacture and supplier was shown as an important factor in scheduling costs reduction.

Ullrich (2013) [3] approached an integrated problem which consists in scheduling a set of jobs in parallel machines with setup dependent times. Already processed jobs are delivery by a vehicle heterogeneous fleet. This author considers processing times and delivery time windows. In order to minimize total tardiness, were presented a MILP model, two classic decomposition approaches and a Genetic Algorithm.

Condotta, Knust, Meier, and Shakhlevich (2013) [4] studied the problem of minimizing total tardiness in a linear machine and delivery costs, considering due dates and limited fleet of identical vehicles. In this work, were developed a

MILP model, a Tabu Search algorithm for partial solutions and then delivery was calculated in polynomial time for partial solutions.

Xiao, Konak (2015) [5] presented the Green Vehicle Routing and Scheduling Problem (GVRSP) considering traffic dependent time conditions with multiple objective functions, such as total CO_2 emission reduction, total traveled distance and time. A new mathematical formulation is proposed, plus a Simulated Annealing (SA) algorithm for large instances. Results shown that up to 50% CO_2 could be reached, with average decreases of 12% and 28% compared to traditional routing methods.

Cheng, Leung and Li (2015) [6] studied the integrated job scheduling and delivery problem considering batch processing in a machine, respecting its capacity. In their work, there are arbitrary job sizes and processing times, and vehicles have same capacity. It was generated a lower bound for ideal total cost and an Ant Colony algorithm, obtaining good results in a reasonable time for large instances.

Billaut, Croce and Ta (2017) [7] approached the integrated job scheduling and vehicle routing problem in a Flow Shop environment with only one vehicle available. It was minimized the total tardiness by implementing two Tabu Search algorithms and a matheuristic. Experiments were realized in random data sets.

Wang, Yao, Sheng and Yang (2019) [8] studied the integrated job scheduling and vehicle routing problem in a linear machine and multiple vehicles. In this environment, it is possible to change a machine between two adjacent jobs. The main objective is to minimize total carbon emission and later were added minimizing total costs. In their work were developed a MILP model and a hibrid Tabu Search algorithm.

Ghannadpour and Zarrabi (2019) [9] approached the integrated multi-objective heterogeneous fleet and production problem, minimizing energy following sustainable requirements. There are two minimization cases: minimizing the number of rental vehicles, customer satisfaction and used energy (first case) and minimizing total traveled distance (second case). It presents a new mathematical formulation for the Vehicle Routing problem with time windows (VRPTW) and a new solution based on an evolutionary algorithm.

Liu, Wenli and Kunpeng Li, and Zou (2020) [10] studied the integrated job scheduling and vehicle routing problem with a linear machine, which main objective was to minimize total delivery times. In this work, there is a limited and identical vehicle fleet. It was proposed a Variable Neighborhood Search (VNS) plus a Tabu Search algorithm for intensification. Results were compared to CPLEX Solver and two different literature heuristic algorithms, obtaining good results.

Tamannaei and Morteza (2019) [11] approached the integrated job scheduling and vehicle routing problem minimizing total weighted tardiness and delivery costs. In their work, jobs have same sizes and vehicles the same cost and capacity. In their work, it was proposed a MILP model, a Genetic Algorithm metaheuristic with three different strategies and a exact Branch-and-Bound (B&B) method.

3 Problem Description

In this section, the integrated problem is better described along with it's parameters, indices and decision variables.

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.

Consider a linear machine handling N jobs, where each job i with size s_i must be processed in sequence after the production beginning ($t = 0$). A customer order only one job. In case job i is processed before other job j, the variable A_ij will have value '1', or '0' otherwise. The completion time C_i of a job i is its accumulated processing time after machine begins. To deliver jobs, there are K available vehicles, each one of used vehicles leaving origin 'O' and returning to it in route end. In a route, in case of a job i is delivered immediately before other job j by a vehicle k, the variable X_{ij}^k will receive value '1', or '0' otherwise. The travel distance between two customers i and j is represented by t_{ij}, with $i = 0$ or $j = 0$ indicating origin.

Each vehicle k has capacity Q_k, utilization cost F_k and start time indicated by S_k. If a vehicle k is used, the variable Y_k will assume a value '1', or '0' otherwise.

Each job i has its own processing time P_i and due date d_i attached, begin penalized by a constant w_i multiplied by its tardiness T_i. The destination time is represented by D_i. The tardiness T is given by $\max(D_i - d_i, 0)$, the difference between its delivery time and its due date.

The multi-objective function defined by minimizing total traveled distance, total weighted tardiness and vehicles cost is given below:

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

4 Mathematical Model

In this section, Mixed Integer Linear Programming (MILP) is proposed. This model is an adapted version of (Tamannaei and Rasti-Barzoki, 2019) [11] model, which considers same vehicle capacities Q and utilization cost F. The model is described as follows:

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

st.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The multi-objective function (1) minimizes the total travel distance, total job weighted tardiness and the used vehicles cost. 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) define 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 start vehicle time S_k.

Constraints (12) define 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 Solution Representation

The solution representation is disposed linearly, where each job and vehicle is labeled with its own ID number. The set of N jobs will be subdivided into K subsets, and each subset assigned to a different vehicle. For each vehicle k, its start time S_k is given by the sum of accumulated past jobs processing times P (after $t = 0$).

In each subset will be defined a delivery route. Considering S a feasible solution, a vehicle $k \in K$ and R_k its route, the path representation is given by:

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

In this context, Ji are the delivered jobs in route $\forall i = 1, ...|N_k| \in N$, and $|N_k|$ indicates the amount of carried jobs. Every vehicle starts from origin 'O' and must return to it, visiting each customer only once. Figure 1 describes visually the solution representation.

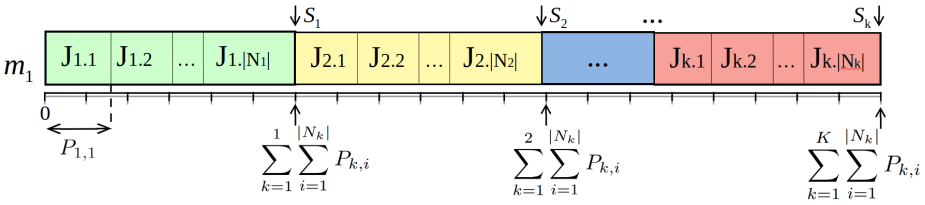


Fig. 1. Visual solution representation for the integrated problem.

A job subset may be empty and still be assigned to a vehicle k, indicating that this vehicle isn't in use and then $|N_k| = 0$.

5.1 Numerical Example

A hipothetical case of network transportation is described below, in which origin and six customers (C_1 à C_6) are disposed, being the distante between each of them euclidian.

Jobs and vehicle fleet are detailed in Table 2. The jobs information describes its priority and urgency via penalty weights and due dates. The heterogeneous vehicles (V_1, V_2 e V_3) compose the avaiable fleet.

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

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			

Figure 2 describes in a visual way the optimal solution. In this example, all three vehicles were used for job delivery, having start times $S_1 = 180$, $S_2 = 301$ e $S_3 = 88$. The percentage in each line end represents the space occupied by carried jobs (100% means that a vehicle is full). Along all six jobs, just J_6 was delivered without any tardiness.

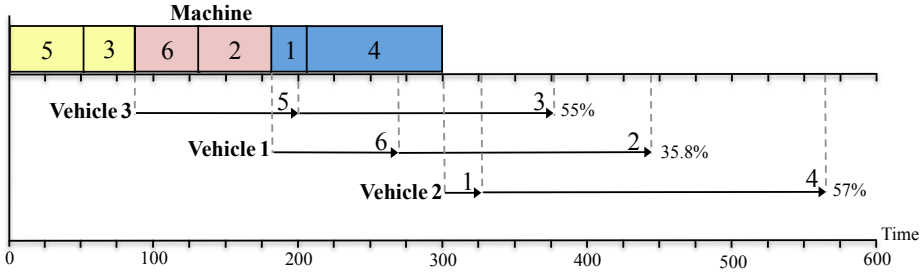


Fig. 2. Visual solution description in a Gantt Chart.

In this optimal solution, job completion times C are, respectively: 276, 180, 88, 301, 52 and 137.

The destination times D, which indicate the exact delivery moment after machine begins, are: 328, 445, 378, 565, 200 e 270.

The job tardiness T, indicated by the difference between destination time and its due date were: 62, 98, 75, 134, 23 e 0.

The optimal objective function value of this instance is 6127.5, being composed by the three minimization objectives. The first one is given by the three vehicles' travel costs, which are: $DC = (112 + 178 + 253) + (90 + 175 + 260) + (27 + 237 + 254) = 1586$.

The second one is defined by the vehicle utilization costs, and in this case all three vehicles were used, then $VC = 1224 + 1116 + 960 = 3300$.

The third and last one indicates the total weighted costs, which is the sum of all jobs tardiness T multiplied by its penalty weights w. The total weighted costs are: $TC = (62 * 1.8) + (98 * 2.3) + (75 * 4.3) + (134 * 4) + (23 * 2) + (0 * 4.6) = 1241.5$.

Thus, the sum of all objective function parts gives the total optimal value: $DC + VC + TC = 6127.5$.

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:

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

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:

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

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:

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

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

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

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 Busca Local e Vizinhanças

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.

7.1 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.

7.2 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.

7.3 Abordagens de RVND

O processo de busca local é realizado por VND (Mladenovic and Hansen 1997) [12], 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).

Algorithm 1: RVND(S)

```
   Input  : Solution S
   Output : Better or equal S* solution
 1 S* ← S;
 2 NeighborhoodSet ←Random Order(Neighborhoods);
 3 for i ← 1 in NeighborhoodSet do
 4 |   currNeighborhood ← NeighborhoodSet[i]; // Select Neighborhood
 5 |   S ← currNeighborhood(S); // Apply Local Search
 6 |   if f(S) ≤ f(S*) then
 7 |   |   S* ← S;
 8 |   |   NeighborhoodSet ←Random Order(Neighborhoods);
 9 |   |   j ← 1; // Restart Local Search
10 |   end
11 end
   Output : S*
```

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.

A segunda versão, RVND-Custom, é baseada na técnica de aprimoramento interno (Penna and Ochi, 2013) [15], 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).

Algorithm 2: RVND-Custom(S)

```
   Input  : Solution S
   Output : Better or equal S* solution
 1 S* ← S;
 2 InterRoute ←Random Order(Inter-Route Neighborhoods);
 3 for i ← 1 in InterRoute do
 4 |   currInter ← InterRoute[i]; // Get Inter-Route Neighborhood
 5 |   S' ← currInter(S); // Apply Inter Local Search
 6 |   if f(S') ≤ f(S) then
 7 |   |   IntraRoute ←Intra-Route Neighborhoods;
 8 |   |   for j ← 1 in IntraRoute do
 9 |   |   |   currIntra ← IntraRoute[j];// Apply Intra Local Search
10 |   |   |   S'' ← currIntra(S');
11 |   |   |   if f(S'') ≤ f(S') then
12 |   |   |   |   S' ← S'';
13 |   |   |   |   j ← 1; // Restart Intra-Route Socal Search
14 |   |   |   end
15 |   |   end
16 |   |   i ← 1;
17 |   |   InterRoute ←Random Order(Inter-Route Neighborhoods);
18 |   end
19 |   if f(S') ≤ f(S*) then
20 |   |   S* ← S';
21 |   end
22 end
   Output : S*
```

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 encon-

trada é atualizada a cada iteração da busca externa (Passos 19-21). Ao final do algoritmo, a melhor solução $S*$ é retornada.

8 Abordagens de Solução

Neste trabalho são propostos três algoritmos para solucionar o Problema de Sequenciamento de Produção e Entrega: dois algoritmos de Iterated Local Search (ILS) e um Algoritmo Genético (G.A.).

O primeiro algoritmo de Iterated Local utiliza a implementação básica de RVND e o segundo utiliza o modelo de RVND customizado com a técnica de aprimoramento interno de rotas (Penna and Ochi, 2013 [15]). O Algoritmo Genético (GA) desenvolvido contém método de Torneio Binário e mutação em 2-point Crossover.

8.1 Soluções Iniciais

A qualidade de um ótimo local obtido por método de busca local depende da solução inicial trabalhada (El-Ghazali Talbi, 2009) [13]. 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) [14].

$$ATC_i = \frac{w_i}{P_i} * \exp(-\frac{max(d_i - P_i - t, 0)}{\frac{\sum_{i=1}^{N} P_i}{N}}) \quad (14)$$

$$ATC_i = \frac{w_i}{P_i} * \exp(-\frac{max(d_i - P_i - t, 0)}{\overline{P}}) \quad (15)$$

$$ATC_i = \frac{w_i}{P_i} * \exp(-max(d_i - P_i - t, 0)/\overline{P}) \quad (16)$$

$$WMDD_i = \frac{1}{w_i} * max(P_i, d_i - t)) \quad (17)$$

$$WEDD_i = \frac{d_i}{w_i} \quad (18)$$

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.

9 Iterated Local Search

Ao gerar variados ótimos locais, utilizar a Iterated Local Search torna possível obter aprimoramento da qualidade da solução (Talbi, 2009) [13]. 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'.

9.1 ILS-RVND

O esquema geral da metaheurística ILS proposta é apresentado em Algoritmo 4. 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 8.1, 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 α 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* ← Initialize(S*);
 2 for i ← 1 to MaxIter do
 3 |   S ← GenerateInitialSolution();
 4 |   S' ← RVND(s);
 5 |   for j ← 1 to MaxIterILS do
 6 |   |   if f(S) ≤ f(S') then
 7 |   |   |   S' ← S;
 8 |   |   |   j ← 1; // Restart Perturbing Process
 9 |   |   end
10 |   |   S' ←Perturb(S', α);
11 |   |   S' ←RVND(S'); // Local Search on Perturbed S'
12 |   end
13 |   if f(S') ≤ f(S*) then
14 |   |   S* ← S';
15 |   end
16 end
   Output : S*
```

Algorithm 4: ILS-RVND $(\alpha, MaxIter, MaxIterILS)$

```
   Input  : Parameters of max. iterations numbers
   Output : Best solution S*
 1 S* ← ∞;
 2 for i ← 1 to MaxIter do
 3 |   S' ← Initial-Solution();
 4 |   for j ← 1 to MaxIterILS do
 5 |   |   S' ←RVND(S');
 6 |   |   if f(S') < f(S*) then
 7 |   |   |   S* ← S';
 8 |   |   |   j ← 1; // Restart Perturbing Process
 9 |   |   end
10 |   |   S' ←Perturb(S', α);
11 |   end
12 |   if f(S') < f(S*) then
13 |   |   S* ← S';
14 |   end
15 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 $S*$.

9.2 ILS-RVND com Aprimoramento de Rota Interna

Penna and Ochi, 2013 [15] 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 4 ao realizar a substituição referida em Figura 3.

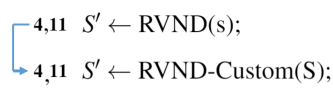


Fig. 3. Mudança entre acionamentos de RVND.

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.

10 Algoritmo Genético

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) [13]. 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 5.

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 6. 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).

10.1 Busca Local Rápida

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 6.

Algorithm 5: Genetic Algorithm ($PopSize, MaxIter$)

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

$P \leftarrow$ Generate random population($PopSize$);
$P \leftarrow$ RVND(P); `// Apply local search to new population P`
$P* \leftarrow$ Best(P); `// Save best solution of P`
for $i \leftarrow 1$ to $MaxIter$ **do**
  $P' \leftarrow \emptyset$;
  for $j \leftarrow 1$ to ($PopSize - 1$) **do**
    $Parent1 \leftarrow$ Binary Tournament(random(P), random(P));
    $Parent2 \leftarrow$ Binary Tournament(random(P), random(P));
    $Offspring1 \leftarrow$ Crossover($first, second$);
    $Offspring2 \leftarrow$ Crossover($first, second$);
    if Feasible($Offspring1$) and Feasible($Offspring2$) **then**
      $P' \leftarrow P'$ + BinaryTournament($Offspring1, Offspring2$);
    else
      if Feasible($Offspring1$) or Feasible($Offspring2$) **then**
        $P' \leftarrow P'$ + Feasible from($Offspring1, Offspring2$);
      else
        if !Feasible($Offspring1$) and !Feasible($Offspring2$) **then**
          $P' \leftarrow P'$ + Mutation(random($Parent1, Parent2$));
        end
      end
    end
  end
  $P' \leftarrow$ P' + P*; `// Include best solution to P'`
  $P' \leftarrow$ Fast Local Search(P'); `// Fast Local Search in population P'`
  if f(Best(P')) $\leq$ f($P*$) **then**
    $P* \leftarrow Best$(P');
  end
  $P \leftarrow P'$;
end
Output : $P*$

Algorithm 6: Fast Local Search(S)

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

$S* \leftarrow S$;
$NeighborhoodSet \leftarrow$*Inter-Route* and *Intra-Route Neighborhoods*;
for $i \leftarrow 1$ to $NeighborhoodSet$ **do**
  $currNeighborhood \leftarrow NeighborhoodSet[i]$;
  $S' \leftarrow$ currNeighborhood(S);
  if f(S') $\leq$ f($S*$) **then**
    $S* \leftarrow S'$;
  end
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.

11 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 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:

$$RPD = \frac{f_{method} - f_{best}}{f_{best}} * 100\% \tag{19}$$

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.

11.1 Calibração de Parâmetros

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 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, sendo cada uma delas executada 10 vezes para as 1740 instâncias.

Os resultados foram analisados por meio do teste não-paramétrico de Kruskall-Wallis utilizando o Average RPD(%) como variável de resposta. Seguindo a Equação 19, 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 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.

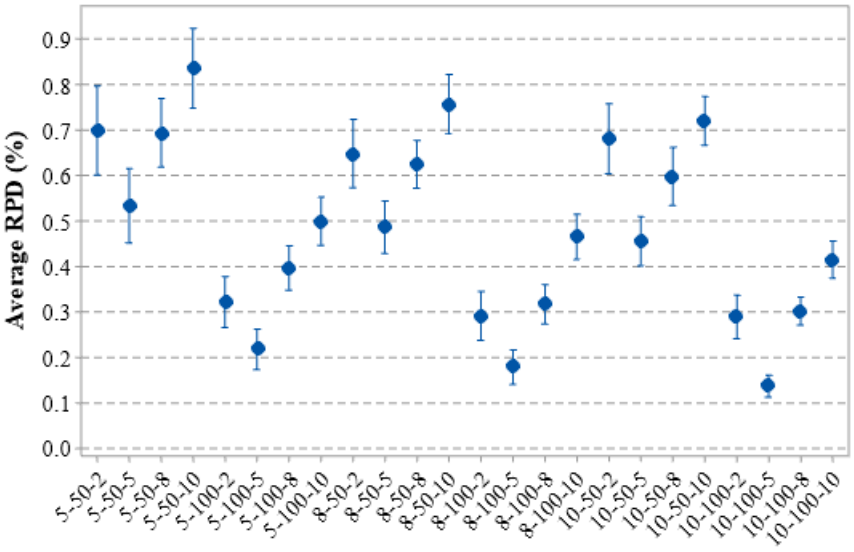


Fig. 4. Mean plot and confidence intervals at 95% confidence level for ILS_RVND_1 experiment.

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 Kruskall-Wallis, o p-valor=0.00 $\leq$ 0.05 obtido indica que a hipótese nula deve ser rejeitada, concluindo assim que pelo menos uma implentação 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 possuiam menor o número de perturbações α 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 significamente diferente. Isto posto, foi definida a combinação '10-100-2' para o algoritmo.

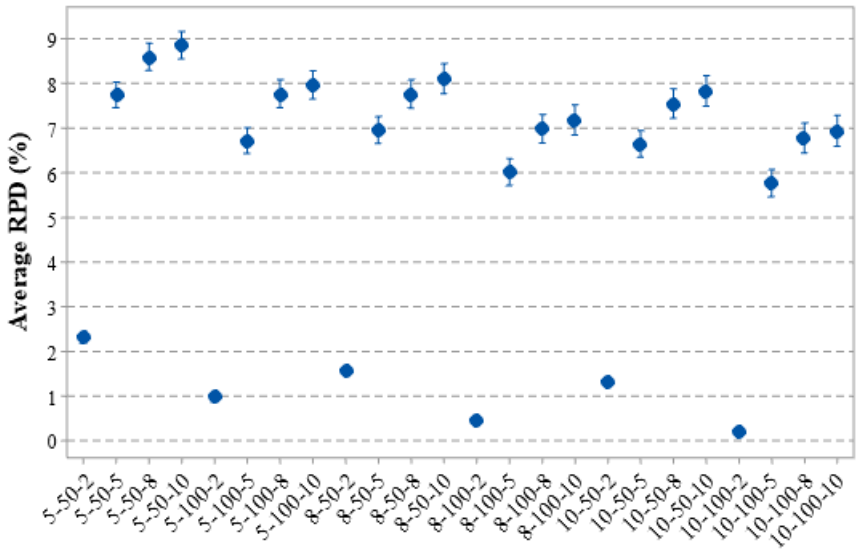


Fig. 5. Mean plot and confidence intervals at 95% confidence level for ILS_RVND_2 experiment.

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 Kruskall-Wallis, o p-valor=0.00 $\leq$ 0.05 obtido 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 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.

11.2 Resultados em Instâncias Pequenas

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.

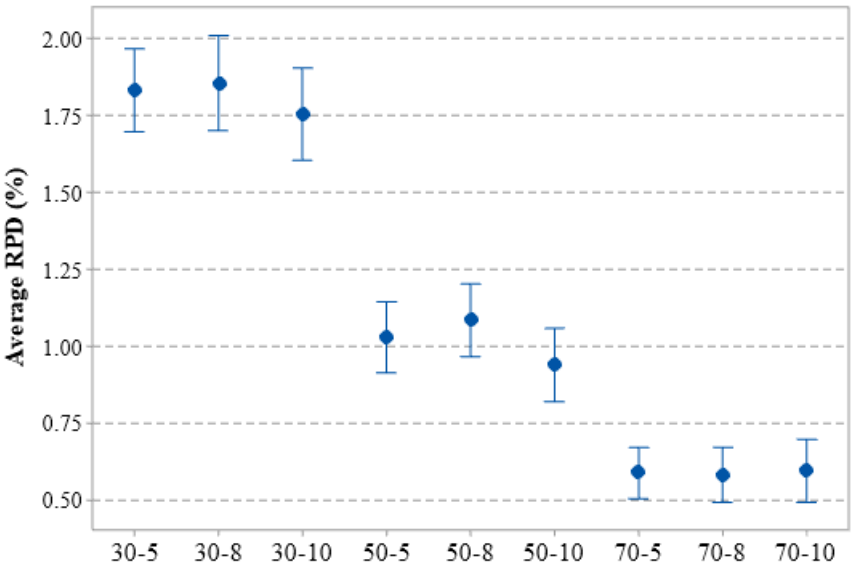


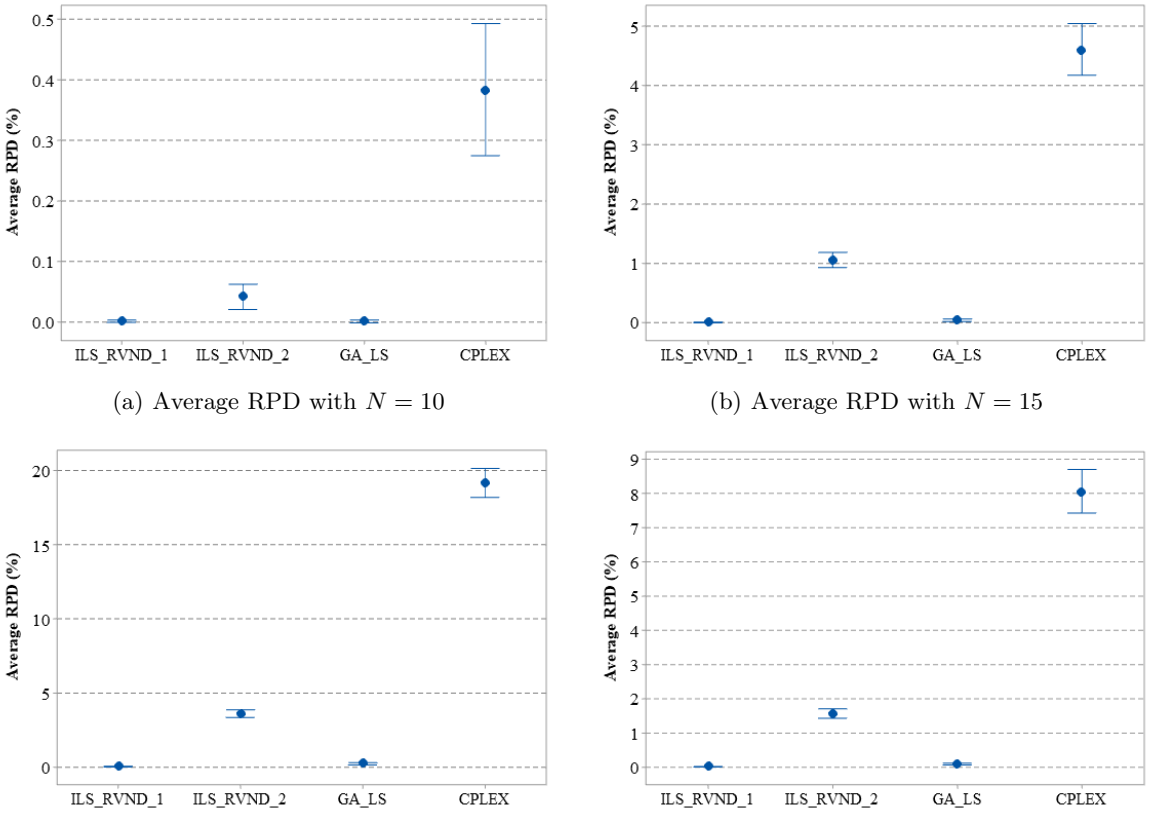
Fig. 6. Mean plot and confidence intervals at 95% confidence level for GA_LS experiment.

A performance de cada algoritmo é medida pelo Average RPD(%) seguindo a Equação 19, 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 Kruskall-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 valores de Average RPD(%) das instâncias agrupadas pelo número de tarefas, e por último um gráfico geral de todas intâncias.

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_RNVD_1 possui a menor média geral. Este algoritmo disputa os melhores resultados com o Algoritmo Genético GA_LS em



(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.

66,67% dos grupos, e possui a média para o Best RPD(%) quando comparada à heurística GA_LS para $N = 20$ e $K = 6$.

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
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

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 médio foi alcançado pela heurística populacional (GA_LS). Dentre as heurísticas de solução única, ILS_RVND_1 obteve o melhor tempo médio de execução.

11.3 Resultados em Instâncias Grandes

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 19 e plotadas na Figura 11.3 com intervalos de à 95% de confiança. Neste grupo de instâncias, é possível observar um distanciamento considerável do algoritmo ILS_RNVD_2 para os demais algoritmos.

Pelo Teste de Kruskall-Wallis, o p-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

Table 4. Runtime Average Analysis over small instances.

	ILS_RVND_1	ILS_RVND_2	GA_LS	CPLEX
N x K	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

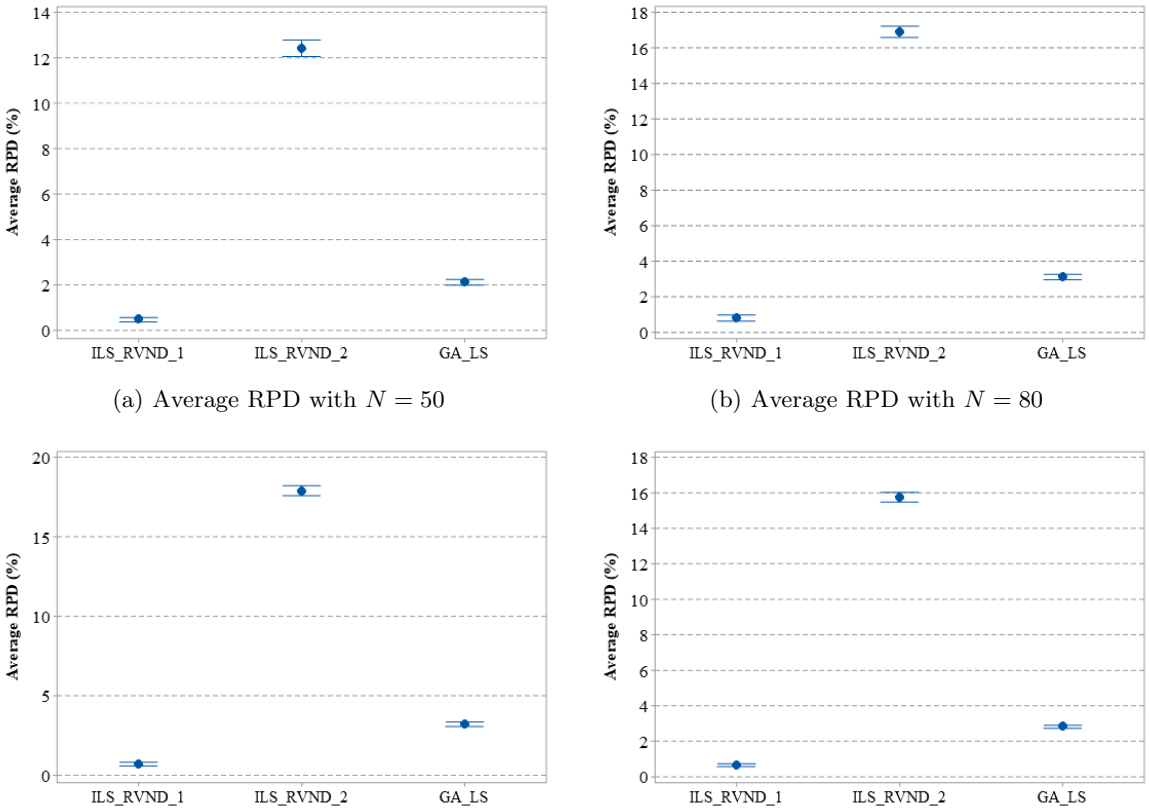
para $N \in \{50, 80, 100\}$, o que indica que todas os resultados são realmente estatisticamente diferentes.

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.

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.

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 obtêr um número favorável de melhoras a assim acionar o aprimoramento interno. Isso não acontece na primeira heurística ILS_RVND_2, 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).



(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.

Table 5. Best RPD and Avg. RPD analysis 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	0.89	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

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

12 Conclusões

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 havia sido abordada na literatura esta versão do 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 Solver CPLEX em um tempo muito inferior. O Solver CPLEX foi executado para instâncias de até 20 clientes. 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 com 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.

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.

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.

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. 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.

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.