HEURÍSTICAS PARA O SEQUENCIAMENTO DA PRODUÇÃO E ROTEAMENTO DE VEÍCULOS COM FROTA HETEROGÊNEA

RESUMO

Neste artigo aborda-se o problema integrado de sequenciamento de tarefas numa máquina simples e o roteamento de veículos com frota heterogênea. Neste problema, um conjunto de tarefas devem ser processados numa máquina e, utilizando um conjunto de veículos, as tarefas processadas devem ser entregues aos respetivos clientes. O objetivo do problema é determinar o sequenciamento das tarefas na máquina e as rotas dos veículos utilizados para fazer as entregas, de tal forma que seja minimizada a soma do atraso total ponderado das tarefas, o custo total dos veículos utilizados e o tempo total das viagens realizadas. Para resolver o problema, inicialmente propõe-se um modelo de programação linear inteira mista. Motivado pela complexidade computacional do problema, são propostas duas heurísticas híbrida baseada nas meta-heurísticas Iterated Local Search (ILS) e Random Variable Neighborhood Descent (RVND). Os desempenhos das heurísticas propostas são avaliados e comparados através de experimentos computacionais em conjunto de instâncias geradas aleatoriamente. Os resultados mostram que uma das heurísticas propostas possui um desempenho superior, como pode ser visto através da comparação com um Algoritmo Genético da literatura.

PALAVRAS CHAVE. Sequenciamento. Roteamento de Veículos. Meta-heurísticas.

Tópicos: Metaheurísticas, Otimização Combinatória.

ABSTRACT

In this article we address the integrate single machine scheduling problem and the heterogeneous fleet vehicle routing problem. In this problem, a set of jobs must be processed on a machine and, using a set of vehicles, the jobs must be delivered to the respective customers. The objective of the problem is to determine the sequencing of jobs on the machine and the routes of the used vehicles, in order to minimize the sum of the total travel costs of vehicles, the total fixed costs of vehicles, and the total penalty costs for delivery tardiness. To solve the problem, initially a mixed integer linear programming model is proposed. Motivated by the computational complexity of the problem, two hybrid heuristics based on the Iterated Local Search (ILS) and Random Variable Neighborhood Descent (RVND) meta-heuristics are proposed. The performances of the proposed heuristics are evaluated and compared using computational experiments on a set of instances randomly generated. The results show that one of the proposed heuristics has a superior performance, as can be seen through the comparison with a genetic algorithm from the literature.

KEYWORDS. Scheduling. Vehicle Routing. Metaheuristics.

Paper topics: Metaheuristics, Combinatorial Optimization.

1. Introdução

Para aumentar a competitividade no mercado global, as empresas industrias 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. Essas atividades incluem a compra de matérias-primas, controle de estoque, produção dos produtos, e entrega dos produtos a os clientes finais dentro dos prazos estabelecidos. Todas essas atividades devem ser muito bem planejadas e otimizadas para que possam ser entregues produtos de qualidade e gerar resultados positivos. As operações de produção e distribuição são ligadas diretamente, sem etapas intermediárias (Chang et al. [2014]). Os objetivos principais 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 a os respectivos clientes por uma frota heterogênea de veículos, levando em consideração 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).

Problemas de programação da produção e roteamento de veículos são dois problemas de otimização combinatória bem estudados na literatura, e geralmente são abordados de forma separada (Zarandi et al. [2020], Braekers et al. [2016]). Na literatura existem poucos trabalhos que abordam a integração da programação da produção e transporte. A seguir apresentam-se alguns trabalhos relacionados.

Ullrich [2013] estudou 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 Programação Linear Inteira Mista (PLIM), duas abordagens clássicas de decomposição e um Algoritmo Genético (AG). Chang et al. [2014] abordaram um problema onde, as tarefas são primeiro processadas em um sistema de máquinas paralelas não relacionadas e depois são distribuídas aos clientes por veículos sem estoque intermediário. Eles propuseram um algoritmo de Colônia de Formigas para minimizar a soma ponderada dos tempos totais de entrega das tarefas e o custo total de distribuição. Karaoğlan e Kesen [2017] estudaram o problema de produção e distribuição com a vida útil do produto. Eles consideraram um único veículo para transporte, onde o o veículo pode realizar várias viagens. Para determinar o tempo mínimo necessário para produzir e entregar todas as demandas dos clientes, eles propuseram um algoritmo Branch-and-Cut. Zou et al. [2018] abordaram o problema integrado de programação de produção e roteamento de veículos em um ambiente de produção sob encomenda, com o objetivo de minimizar a produção e propor um AG. Tamannaei e Rasti-Barzoki [2019] abordaram um problema integrado de sequenciamento da produção em uma máquina e roteamento de veículos com o objetivo de minimizar a soma do atraso total ponderado e dos custos de transporte, incluindo o custo fixo do veículo e o custo da viagem na rede de transporte. Estes autores consideraram uma frota de veículos homogêneos, tarefas de tamanhos iguais e propuseram um modelo PLIM, um algoritmo Branch-and-Bound e um AG. Recentemente, Liu et al. [2020] estudaram o sequenciamento da produção e roteamento de veículos, de forma integrada, onde há uma única máquina para produção e um número limitado de veículos homogêneos para o transporte. Para minimizar a soma dos tempos de entrega dos pedidos, os autores apresentaram um modelo PLIM e uma heurística *Variable*

Neighborhood Search (VNS).

Neste trabalho propõe-se duas heurísticas híbridas com múltiplos reinícios baseadas na meta-heurística *Iterated Local Search* (ILS), onde a busca local do ILS é gerenciada pelo método *Random Variable Neighborhood Descent* (RVND). O RVND utiliza sete estruturas de vizinhanças que sao ordenadas aleatoriamente. As heurísticas são comparadas com um AG proposto por Tamannaei e Rasti-Barzoki [2019]. Para resolver instâncias de pequeno porte do problema em estudo, apresenta-se um modelo PLIM que é resolvido pelo solver CPLEX.

O restante deste artigo está organizado da seguinte maneira. A Seção 2 primeiro define o problema abordado e, em seguida, apresenta o modelo PLIM e um exemplo numérico do problema. As heurísticas propostas e seus componentes estão detalhados na Seção 3. Então, a Seção 4 apresenta os testes experimentais que foram realizados para analisar o desempenho das heurísticas. Finalmente, a Seção 5 apresenta as conclusões do trabalho.

2. Descrição e Modelagem do Problema

O problema abordado neste artigo consiste na integração de dois subproblemas: Sequenciamento de tarefas numa máquina simples e Roteamento de veículos com frota heterogênea. Neste problema integrado, denotado por *SeqRot*, N produtos (tarefas) devem ser produzidos (processados) numa máquina e, utilizando um conjunto de K veículos, as tarefas processadas devem ser entregues a seus respetivos clientes. Para cada tarefa i ($i = 1 \dots N$) são conhecidos os seguintes dados: tempo de processamento P_i, tamanho (espaço que ocupa num veículo) s_i, prazo (ou data) de entrega d_i e penalidade por atraso na entrega w_i. Para cada veículo k são conhecidos, sua capacidade de carga Q_k e seu custo de utilização F_k. Considera-se que as tarefas são produzidas numa fábrica que representa o depósito ($i = 0$), e os locais de entrega de cada tarefa estão espalhadas numa área geográfica, de tal maneira que os tempos de viagem (ou distâncias) entre cada par locais são conhecidos. O tempo de viagem do local i para o local j é denotado por t_{ij}, sendo que $t_{ij} = t_{ji}$, $\forall i,j = 0 \dots N$.

O objetivo do problema *SeqRot* é determinar a ordem de processamento das tarefas (sequenciamento das tarefas na máquina) e determinar as rotas dos veículos utilizados para fazer as entregas, de tal forma que seja minimizada a soma do atraso total ponderado das tarefas, custo total dos veículos utilizados e tempo total das viagens realizadas. Todas as tarefas estão disponíveis para serem processadas a partir de um tempo inicial (tempo zero), e a máquina só pode processar uma tarefa de cada vez. No problema *SeqRot*, também devem ser determinadas os instantes de partida (saída da fábrica) dos veículos. Um veículo só poderá sair da fábrica após finalizar o processamento de todas as tarefas que serão transportadas no veículo, ou seja, o subproblema de roteamento de veículos aqui abordado considera tempos de disponibilidade, em inglês *realese date* (Archetti et al. [2015]). No problema não são considerados os tempos gastos para fazer a carga e descarga das tarefas. Os veículos sairão da fábrica, visitarão os clientes, uma única vez, e retornarão à fábrica.

Vale ressaltar que o problema *SeqRot* é NP-difícil, pois os subproblemas, sequenciamento de tarefas em uma máquina com a minimização do atraso total e roteamento de veículos com frota heterogênea (RVFH), são problemas NP-difíceis (Du e Leung [1990], Chen [2010]).

2.1. Modelo Matemático

Nesta seção, um modelo matemático baseado em Programação Linear Inteira Mista (PLIM) é apresentado. Este é modelo é uma adaptação do modelo proposto por Tamannaei e Rasti-Barzoki [2019]. Neste trabalho consideram-se veículos com capacidades e custos variáveis, e tarefas com tamanhos diferentes. As variáveis de decisão do modelo são:

A_{ij} Variável binária é igual a '1' se a tarefa i é processada na máquina antes da tarefa j.

C_i Tempo de conclusão do processamento da tarefa i.

Y_k Variável binária igual a '1' se o veículo k é usado, caso contrário a variável é '0'.
S_k Tempo/instante de partida (saída do depósito) do veículo k.
X_{ij}^k Variável binária igual '1' se o veículo k viaja do local i para o local j (neste caso as tarefas i e $j \neq 0$ são designadas ao veículo k), caso contrário a variável é igual a '0'.
D_i Instante de entrega da tarefa i.
T_i Atraso da tarefa i.

O modelo de PLIM é descrito a seguir:

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

s.a.

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

$$\sum_{i=0}^{N}\sum_{j=0,j\neq i}^{N} s_i * X_{ij}^k \leq Q_k * Y_k, \quad \forall k = 1 \dots K; \tag{3}$$

$$\sum_{j=1}^{N} X_{0j}^k = Y_k, \quad \forall k = 1 \dots K; \tag{4}$$

$$\sum_{i=0}^{N} X_{ih}^k = \sum_{j=0}^{N} X_{hj}^k, \quad \forall h = 0 \dots N; \forall k = 1 \dots K \tag{5}$$

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

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

$$C_j = \sum_{i=0,i\neq j}^{N} (P_i * A_{ij}) + P_j, \quad \forall j = 1 \dots N; \tag{8}$$

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

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

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

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

$$T_i \geq 0, \quad D_i \geq 0, \quad S_k \geq 0, \quad \forall i = 1 \dots N; \quad \forall k = 1 \dots K; \tag{13}$$

$$X_{ijk} \in \{0,1\}, \quad A_{ij} \in \{0,1\}, \quad Y_k \in \{0,1\}, \quad \forall i,j = 0 \dots N; \forall k = 1 \dots K; \tag{14}$$

A Equação (1) define a minimização da soma dos custos totais das viagens dos veículos, os custos fixos dos veículos e o atraso total ponderado nas entregas das tarefas. As restrições (2) garantem que cada cliente (dono da uma tarefa) seja visitado exatamente uma vez. As restrições (3) indicam que, se um veículo k é usado ($Y_k = 1$), o tamanho total de todas as tarefas transportadas neste veículo não exceda sua capacidade. As restrições (4) garantem que, se qualquer veículo for usado, ele certamente deixará o depósito, caso não seja usado ele ficará no depósito. As restrições (5) garantem que, se um veículo chega a um cliente, este veículo deve deixar esse cliente. As restrições (6) e (7) determinam um sequenciamento válido das tarefas na máquina (isto é, uma tarefa deve ser processada uma única vez e a máquina só pode processar uma tarefa de cada vez). O modelo

considera uma tarefa fictícia '0' para indicar a antecessora da primeira tarefa da sequencia ou a sucessora da última tarefa. Os tempos de conclusão (de processamento) das tarefas são determinados pelas restrições (8). As restrições (9) afirmam que, se a tarefa j é transportada pelo veículo k, o tempo de partida deste veículo deve ser maior o igual que o tempo de conclusão do trabalho j. As restrições (10) e (11) determinam os instantes de entrega das tarefa aos clientes. As restrições (10), determinam o instante de chegada de um veículo ao primeiro cliente da sua rota. As restrições (11) determinam os instantes de chegada de um veículo aos outros clientes da sua rota. M é um número positivo suficientemente grande, definido como: $M = \sum_{i=1}^{N} P_i + N * max_j\{t_{0j}\}$. As restrições (12) e (13) indicam que o atraso de uma tarefa é calculado como o máximo de 0 e a diferença entre o instante de entrega e a data de entrega prevista. As restrições (13) indicam a não-negatividade das variáveis T_i, D_i e S_k. As restrições (14) indicam que X_{ijk}, A_{ij} e Y_k são variáveis binárias. No modelo são considerados $P_0 = 0$, $C_0 = 0$, $D_0 = 0$ e $T_0 = 0$.

2.2. Exemplo Numérico

Na Figura 1 apresenta-se os dados para uma instâncias do problema com $N = 6$ tarefas (jobs) e $K = 3$ veículos. Cada tarefa i deve ser entregue ao seu respectivo cliente c_i.

Jobs	P_i	d_i	w_i	s_i	Vehicles	Q_k	F_k
1	25	266	1,8	20	1	120	1224
2	49	347	2,3	31	2	110	1116
3	36	303	4,3	25	3	80	960
4	96	431	4,0	86			
5	52	177	2,0	33			
6	43	315	4,6	42			

(a) Dados de entrada para 6 tarefas e 3 veículos.

	Depósito	c_1	c_2	c_3	c_4	c_5	c_6
Depósito	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

(b) Matriz de distâncias t_{ij}.

Figura 1: Exemplo de uma instância do problema.

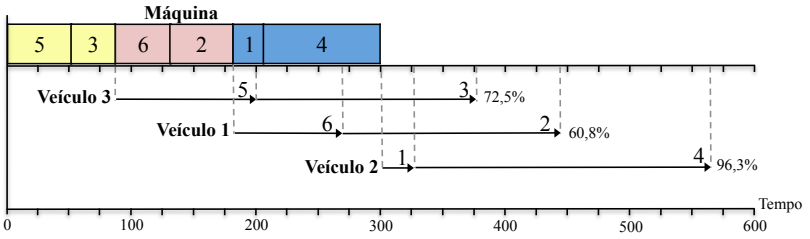


Figura 2: Gráfico de Gantt (sequenciamento e roteamento) para uma instâncias com $n = 6$ e $K = 3$.

A Figura 2 mostra a solução ótima para a instância apresentada na figura 1. Nesta figura observa-se que, a sequencia de processamento das tarefas é $[5, 3, 6, 2, 1, 4]$, e os três veículos foram utilizados para fazer a entrega das tarefas. O veículo 3 é o primeiro a sair do depósito e veículo 2 é o último. Os instantes que os veículos 3, 1 e 2 deixam o depósito são $S_3 = 88$, $S_1 = 180$ e $S_2 = 301$, respectivamente. As rotas de entrega dos veículos 3, 1 e 2 são respectivamente $\{0 \rightarrow 5 \rightarrow 3 \rightarrow 0\}$, $\{0 \rightarrow 6 \rightarrow 2 \rightarrow 0\}$ e $\{0 \rightarrow 1 \rightarrow 4 \rightarrow 0\}$, onde 0 representa o depósito. Nesta solução a porcentagem de ocupação dos veículos 3, 1 e 2 são respectivamente 72,5%, 60,8% e 96,3%. Para todas as tarefas, os tempos de conclusão de processamento, os instantes de entrega, e os atrasos são respectiamente: ($C_1 = 276$, $C_2 = 180$, $C_3 = 88$, $C_4 = 301$, $C_5 = 52$, $C_6 = 137$), ($D_1 = 328$, $D_2 = 445$, $D_3 = 378$, $D_4 = 565$, $D_5 = 200$, $D_6 = 270$) e ($T_1 = 62$, $T_2 = 98$, $T_3 = 75$, $T_4 = 134$, $T_5 = 23$, $T_6 = 0$). Observa-se que, apenas a tarefa 6 foi entregue sem atraso. O custo total da viagem dos três veículos é $(112 + 178 + 253) + (90 + 175 + 260) + (27 + 237 + 254) = 1.586$. O

custo total de utilização dos veículos é $224 + 1116 + 960 = 3.300$. E, o atraso total ponderado das tarefas é $62*1{,}8+98*2{,}3+75*4{,}3+134*4{,}0+23*2{,}0+0*4{,}6 = 1.241{,}5$. Portanto, ao somar estes três valores obtém-se o valor da função objetivo da solução ótima, $f = 1.586 + 3.300 + 1.241{,}5 = 6.127{,}5$.

3. Heurística Iterated Local Search com Múltiplos Reinícios

A heurística desenvolvida para resolver o problema em estudo é um algoritmo híbrido baseado nas meta-heurísticas Iterated Local Search (ILS) e Random Variable Neighborhood Descent (RVND). A meta-heurística ILS (Lourenço et al. [2003]) é um algoritmo bastante utilizado para resolver problemas de otimização combinatória. Tendo uma solução inicial, este algoritmo repetidamente aplica os procedimentos de perturbação e busca local, até que uma condição de parada seja satisfeita. O procedimento de busca local usa uma ou mais estruturas de vizinhança para modificar sistematicamente a solução atual e encontrar uma melhor. O objetivo da perturbação é escapar do ótimo local, alterando a solução atual de maneira aleatória, para que o processo de busca continue a partir de uma nova solução.

Neste trabalho, o procedimento de busca local do ILS é baseado no RVND. O RVND é um método de descida em vizinhanca variavel (Variable Neighborhood Descent) onde a ordem das vizinhanças é decidida de maneira aleatória. O VND (Hansen e Mladenović [2001]) é uma meta-heurística de busca local que utiliza um conjunto de nv estruturas de vizinhança sequencialmente até que a solução atual se torne ótimo local para cada uma das vizinhanças.

No Algoritmo 1 apresenta-se o pseudocódigo da heurística híbrida com múltiplos reinícios, proposta para o problema *SeqRot*. O algoritmo, chamado de ILS_RVND_1, possui como entrada três parâmetros: $pert$ (nível de perturbação), $NIter_{ILS}$ (número máximo de iterações sem melhora do ILS) e $Nreinicios$ (número de reinícios do algoritmo ILS_RVND_1). O algoritmo ILS inicia (ou reinicia) com uma nova solução s (passo 3). Esta solução inicial s é melhorada com a busca local RVND (passo 4). O ILS iterativamente executa os passos 5 a 10, até que a condição de parada seja satisfeita. Nos passos 6 e 7, a solução atual é, respectivamente, perturbada e melhorada. Nos passos 8 e 9, a melhor solução obtida pelo ILS é atualizada. Note que, o ILS finaliza se a melhor solução s' não é melhorada por $NIter_{ILS}$ iterações consecutivas. Caso exista uma melhora de s', o contador de iterações j do ILS é resetado (passo 10). Finalizado o ILS, a melhor solução global s^* é atualizada (passos 11-12). O ILS_RVND_1 é reiniciado $Nreinicios$ vezes, e como saída é retornada a solução s^*.

Algoritmo 1: ILS_RVND_1 $(pert, NIter_{ILS}, Nreinicios)$

```
1  f(s*) ← ∞;
2  para i ← 1 até Nreinicios faça
3  |   s ← Solução-Inicial();
4  |   s ← RVND_1(s);
5  |   para j ← 1 até NIter_ILS faça
6  |   |   s' ←Perturba(s, pert);
7  |   |   s'' ← RVND_1(s');
8  |   |   se f(s'') < f(s) então
9  |   |   |   s ← s';
10 |   |   |   j ← 1; // Reinício das iterações do ILS
11 |   se f(s) < f(s*) então
12 |   |   s* ← s;
   saída : s*
```

3.1. Representação de uma Solução e Construção de uma Solução Inicial

Uma solução do problema *SeqRot* é representada por um vetor (de tamanho $N + K$) que armazena os índices de todas as tarefas e de todos os veículos. Na primeira posição do vetor, sempre, é armazenado o índice de um veículo. Por exemplo, para $N = 6$ e $K = 3$, a solução mostrada da Figura 2 é representada pelo vetor $[V_3, 5, 3, V_1, 6, 2, V_2, 1, 4]$. Esta representação divide o conjunto de tarefas em K subconjuntos, R_k, sendo que $0 \leq |R_k| \leq K$, $\forall k = 1, \ldots, K$. Se $|R_k| = 0$, o veículo k não é utilizado no transporte de tarefas. Note que, na representação adotada, a ordem de processamento das tarefas é a mesma da ordem de entrega.

Para iniciar a execução do algoritmo ILS_RVND_1 é necessário de uma solução inicial viável. Uma solução é construída utilizando os seguintes passos. 1) Inicialmente, as tarefas são ordenadas utilizando uma regra de prioridade (obtendo uma sequência de tarefas S). 2) Escolhe-se o veículo de menor custo e insere-se neste veículo a primeira tarefa da sequência. 3) Enquanto existir espaço no veículo as próximas tarefas da sequência são inseridas neste veículo. 4) Se ainda existirem tarefas não inseridas, escolhe-se o próximo veículo de menor custo e repetem-se os passos 3) e 4).

Três regras de prioridade podem ser utilizadas (Molina-Sánchez e González-Neira [2016]): Apparent Tardiness Cost (ATC), Weighted Modified Due Date (WMDD) e Weighted Earliest Due Date (WEDD). Os índices de prioridade destas regras são definidas respectivamente nas equações (15), (16) e (17).

$$I_{ATC}(i) = w_i/P_i * \exp[-max(d_i - P_i - t, 0)/\overline{P}] \tag{15}$$

$$I_{WMDD}(i) = max(P_i, d_i - t)/w_i \tag{16}$$

$$I_{WEDD}(i) = d_i/w_i \tag{17}$$

Para construir uma sequencia S de tarefas, a regra ATC funciona da seguinte maneira. Para cada tarefa i, não inserida em S, calcula-se o índice de prioridade $I_{ATC}(i)$, onde $\overline{P}$ representa a média dos tempos de processamento de todas as tarefas, e t é a soma dos tempos de processamento das tarefas já inseridas em S. A tarefa com *maior* índice de prioridade é selecionada para inserir em S. Para determinar a próxima tarefa a ser inserida em S, os índices de prioridade das tarefas não selecionadas são recalculados (regra dinâmica).

A regra WMDD funciona de forma similar à regra ATC. Na regra WMDD sempre seleciona-se a tarefa i com *menor* índice de prioridade $I_{WMDD}(i)$. A regra WEDD não é dinâmica. Para obter a sequencia S, basta ordenar todas as tarefas em ordem crescente da suas prioridades I_{WEDD}.

3.2. Busca Local RVND

O pseudocódico da busca local RVND e apresentado no Algoritmo 2. O Algoritmo, chamado de RVND_1, recebe como entrada a solução s a ser melhorada. Este algoritmo utiliza um conjunto de nv vizinhanças. Inicialmente as vizinhanças são ordenadas de forma aleatória (passo 1). A cada iteração i escolhe-se a i-ésima vizinhança e determina-se a melhor solução s' desta vizinhança. O algoritmo finaliza se não é encontrada nenhuma solução s' melhor que s para todas as nv vizinhanças (ou seja, se s é um ótimo local para todas as vizinhanças). Caso exista melhoria da solução s, ela é atualizada e o contador das vizinhanças é reinicializado (i.e. novamente todas as vizinhanças serão utilizadas em ordem aleatória).

3.3. Estruturas de Vizinhanças

Na busca local RVND_1 são utilizadas sete estruturas de vizinhanças, 3 do tipo Intra-Rota (movimentos de tarefas dentro da rota de um veículo) e 4 do tipo Inter-Rota (movimentos de tarefas entre rotas diferentes). Ressalta-se que, na representação da solução adotada, a ordem de

Algoritmo 2: RVND_1(s)

```
1 Neighborhood[] ←OrdenaAleatoriamenteAsVizinhanças(nv);
  // nv número de vizinhanças
2 para i ← 1 até nv faça
3 |   N_i ← Neighborhood[i]; // Seleciona i-ésima vizinhança
4 |   s' ← Melhor solução de N_i(s);
5 |   se f(s') < f(s) então
6 |   |   s ← s';
7 |   |   Neighborhood[] ←OrdenaAleatoriamenteAsVizinhanças(nv);
8 |   |   i ← 1; // Reinicio das iterações do RVND
  saída : s
```

processamento das tarefas é a mesma da ordem de entrega das tarefas. Então, na busca local são somente consideradas vizinhanças relacionados ao problema de roteamento de veículos.

Vizinhanças Intra-Rota:

Swap-Adj-Job(R) - Troca duas tarefas adjacentes numa rota R.

Insert-Job(R) - Uma tarefa é removida e inserida em outra posição da rota R.

2-opt(R) - Dentro de uma rota R, dois arcos não adjacentes são excluídos e outros dois são adicionados de maneira que uma nova rota R' seja gerada.

Vizinhanças Inter-Rota:

Estas vizinhanças consideram movimentos de tarefas ou veículos que alteram duas ou mais rotas ao mesmo tempo. Estes movimentos podem gerar soluções inviáveis com relação às capacidades dos veículos, porém estas soluções não são consideradas (não são avaliadas). As quatro vizinhanças usadas são:

Swap-Job(R_1, R_2) - Troca tarefas i de R_1 com uma tarefas j de R_2.

Insert-Job(R_1, R_1) - Insere uma tarefa i de R_1 nas demais posições da rota R_2.

Swap-Adj-Vehicles(R_1, R_2) - Troca de posições dois veículos adjacentes carregando consigo seus lotes de tarefas (ou seja, troca de duas rotas R_1 e R_2). Por exemplo, para a solução da Figura 2, representada por $[V_3, 5, 3, V_1, 6, 2, V_2, 1, 4]$, um movimento gerado por esta vizinhança pode ser trocar a rotas dos veículos V_3 e V_1, gerando a solução $[V_1, 6, 2, V_3, 5, 3, V_2, 1, 4]$. Nesta nova solução as tarefas do veículo V_1 serão processadas primeiro na máquina e este veículo saíra primeiro do depósito.

Insert-Vehicle(R) - Insere a rota R de um veículo antes ou depois de outras rotas. Por exemplo, para a solução representada por $[V_3, 5, 3, V_1, 6, 2, V_2, 1, 4]$, um movimento gerado por esta vizinhança pode ser inserir a rota do veículos V_3 depois da rota do veículo V_2, gerando a solução $[V_1, 6, 2, V_2, 1, 4, V_3, 5, 3]$.

3.4. Perturbação

O objetivo do método de perturbação no algoritmo ILS é escapar de uma solução ótima local. Neste trabalho, dois mecanismos de perturbação são utilizados. O primeiro mecanismo consiste em realizar um número $pert$ de trocas aleatórias de tarefas de rotas diferentes. O segundo consiste em fazer $pert$ inserções aleatórias de tarefas, de uma rota para outra rota. Estes mecanismos podem ser considerados como múltiplos movimentos Inter-Rota: **Swap-Job**(R_1, R_2) e **Insert-Job**(R_1, R_2).

3.5. ILS_RVND_2

Neste trabalho, um segundo algoritmo ILS_RVND com múltiplos reinícios, chamado de ILS_RVND_2, é desenvolvido. A busca local deste algoritmo é baseada num procedimento RVND

similar ao proposto por Penna et al. [2013]. Nesta busca local, as estruturas de vizinhanças Intra-Rota são utilizadas somente se a solução corrente for melhorada utilizando uma vizinhança Inter-Rota. O pseudocódigo desta busca local (denominada RVND_2) é apresentado no Algoritmo 3. Este algoritmo recebe como entrada a solução s a ser melhorada. Inicialmente o conjunto de $nv_1 = 4$ vizinhanças Inter-Rota é ordenado de forma aleatória (passo 1). A cada iteração i determina-se a melhor solução s' da i-ésima vizinhança Inter-Rota (passo 4). Se s' supera a solução s, atualiza-se a melhor solução corrente (passo 6), executa-se uma busca local utilizando as $nv_2 = 3$ vizinhanças Intra-Rota (passos 7-13) e o contador das vizinhanças Inter-Rota é reinicializado para novamente utilizar todas as vizinhanças Inter-Rota. A busca local Intra-Rota finaliza se a solução s não é melhorada com nenhuma das vizinhanças Intra-Rota.

Algoritmo 3: RVND_2(s)

```
1  InterRota[] ←OrdenaAleatoriamenteAsVizinhançasInterRota(nv1);
2  para i ← 1 até nv1 faça
3      N_i ← InterRota[i]; // Seleciona i-ésima vizinhança Inter-Rota
4      s' ← Melhor solução de N_i(s);
5      se f(s') < f(s) então
6          s ← s';
7          IntraRota[] ←OrdenaAleatoriamenteAsVizinhançasIntraRota(nv2);
8          para j ← 1 até nv2 faça
9              N_r ← Escolha aleatoriamente uma vizinhança do conjunto IntraRota[j..nv2];
10             s'' ← Melhor solução de N_r(s);
11             se f(s'') < f(s) então
12                 s ← s'';
13                 j ← 1;
14         i ← 1;
15         InterRota[] ←OrdenaAleatoriamenteAsVizinhançasInterRota(nv1);
   saída : s
```

4. Experimentos Computacionais

Nesta Seção são apresentados os experimentos realizados com os algoritmos ILS_RVND_1 e ILS_RVND_2. Os resultados destes algoritmos são comparados com o software CPLEX que resolve o modelo PLIM do problema (para instâncias pequenas) e com um algoritmo genético (denotado por GA_LS) apresentado por Tamannaei e Rasti-Barzoki [2019]. Todos os algoritmos foram codificados em C++ e executados em um computador Intel(R) Core TM i7-4790K CPU @ 4.00GHz x 8 com 32GB RAM, rodando Ubuntu 18.1 64 bits.

A métrica utilizada para comparação dos algoritmos é o Desvio Percentual Relativo (*Relative Percentage Deviation*) com relação à melhor solução conhecida. Esta métrica é definida como: $RPD = \frac{f_{method} - f_{best}}{f_{best}} * 100\%$, onde f_{best} é o melhor resultado conhecido (encontrado por todos os métodos comparados) e f_{method} é o resultado obtido pelo método avaliado.

4.1. Geração de Instâncias

Para analisar o desempenho dos algoritmos, neste trabalho, foram gerados dois conjuntos de instâncias do problema *SeqRot*. O primeiro conjunto contém instâncias de pequeno porte, onde o número de tarefas N e o número de veículos K são: $N \in \{8, 10, 15, 20\}$, $K \in \{3, 4, 5, 6\}$. O segundo conjunto contém instâncias de médio-grande porte, onde os valores N e K são: $N \in \{50, 80, 100\}$, $K \in \{5, 8, 10, 12\}$

Os parametros das instâncias foram gerados baseado no trabalho de Ullrich [2013]. Para cada tarefa i ($i = 1, \ldots, N$), seu tempo de processamento P_i, tamanho s_i e penalidade de entrega w_i são gerados aleatoriamente (com distribuição uniforme) nos intervalos $U[1, \rho]$, $U[1, P_i]$ e

$U[1{,}0, 5{,}0]$, respectivamente, onde $\rho = 100$. Note que, quanto maior o tempo de processamento de uma tarefa, maior será a probabilidade da tarefa ocupar um maior espaço no veículo.

A data de entrega d_i de uma tarefa i é gerada aleatoriamente no intervalo $U[a+\pi_1, a+\pi_2]$, onde $a = P_i + t_{0i}$. π_1 e π_2 são valores gerados aleatoriamente nos intervalos $U[0, \frac{\sum_{j=1}^{N} P_j}{K+1}]$ e $U[0, \delta * \rho]$, respectivamente. ρ é um parâmetro usado para gerar datas apertadas e folgadas. Para este parâmetro foram utilizados os seguintes valores: $\delta \in \{0{,}5; 1{,}0; 1{,}5; 2{,}0; 2{,}5\}$.

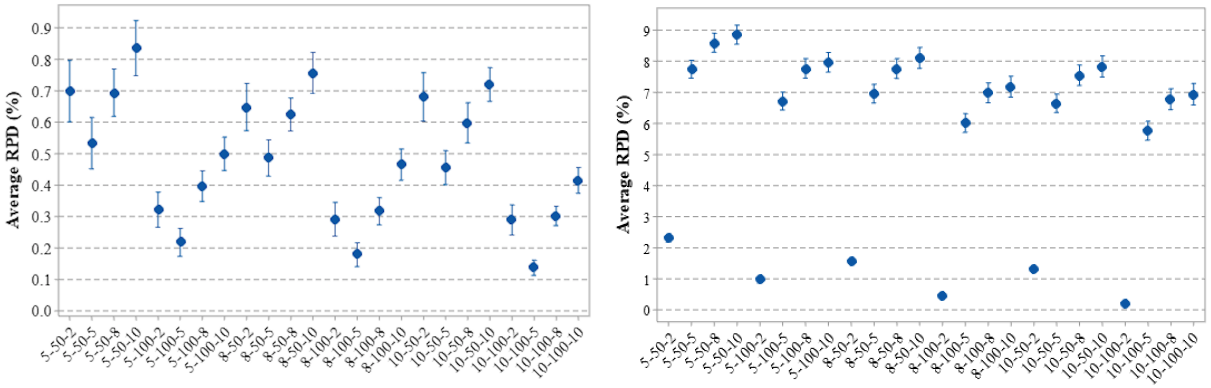
A capacidade de um veículo é gerada de forma que a tarefa de maior tamanho possa ser transportada em qualquer veículo, e todas as tarefas possam ser transportadas utilizando os veículos disponíveis. Para cada veículo k, sua capacidade Q_k é definida como $Q_k = max_{i=1}^{N}\{s_i\} + \tau$. O valor da variável τ é gerando aleatoriamente no intervalo $U[a, \mu * a]$, onde $a = \frac{\sum_{i=1}^{N} s_i}{K}$ e μ é um parâmetro utilizado para variar as capacidades dos veículos. Para instâncias de pequeno porte são utilizados os valores $\mu \in \{1{,}0; 1{,}5; 2{,}0\}$. Para instâncias de médio-grande porte são utilizados os valores $\mu \in \{1{,}5; 2{,}0; 2{,}5\}$.

O custo F_k de um veículo k é diretamente proporcional à sua capacidade: $F_k = N * Q_k$. Para gerar a matriz das distâncias t_{ij} (ou tempos de viagem), as localizações do depósito e dos clientes são distribuídas aleatoriamente em uma área do plano $\mathbb{R}^2$. Primeiro, as coordenadas do depósito são geradas, em seguida as coordenadas dos clientes são geradas ao redor do depósito de tal forma que a distância máxima do depósito até um cliente seja menor ou igual que $\rho * K$. Assim, t_{ij} representa a distância euclidiana do local i para j, $\forall i, j = 0, \ldots, N$, onde $t_{ii} = 0$ e $t_{ij} = t_{ji}$.

Definidos os parâmetros das instâncias, para cada combinação de valores de N, K, δ e μ, foram geradas cinco instâncias de pequeno porte e três instâncias de médio-grande porte. Portanto, foram geradas um total de 4*4*5*3*5 = 1200 instâncias de pequeno porte e 3*4*5*3*3 = 540 instâncias de médio-grande porte.

4.2. Calibração de Parâmetros

Para obter um melhor desempenho dos algoritmos ILS_RVND, é desejável determinar os valores adequados do seus parâmetros. Para calibrar os algoritmos, um novo conjunto de 180 instâncias de médio-grande porte é gerado. O algoritmo, ILS_RVND_1, utiliza três parâmetros: o nível de perturbação ($pert$), o número máximo de iterações sem melhora do ILS ($NIter_{ILS}$) e o número de reinícios do algoritmo ($Nreinicios$). Após realizar testes preliminares, os seguintes valores foram testados para os parâmetros: $Nreinicios \in \{5, 8, 10\}$, $NIter_{ILS} \in \{50, 100\}$ e $pert \in \{2, 5, 8, 10\}$. Combinando os valores destes parâmetros, 24 versões do algoritmo ILS_RVND_1 foram geradas. Para cada combinação e para cada instância, o algoritmo foi executado 10 vezes e a qualidade das soluções é medida pela métrica RPD. Os resultados do experimento são analisados por meio do teste estatístico não-paramétrico de Kruskal-Wallis. Na Figura 3(a) mostra-se o resultado da calibração do algoritmo ILS_RVND_1 para cada 3-tupla $Nreinicios$-$NIter_{ILS}$-$pert$. Esta figura apresenta o gráfico de médias e os intervalos de confiança de Tukey da diferença honestamente significativa (*Honestly Significant Difference* - HSD) com nível de confiança de 95% do teste estatístico. Na figura, os intervalos sobrepostos indicam que não existe diferença estatisticamente significante entre as médias. Os resultados deste teste indicaram que existe diferença estatisticamente significante entre os resultados obtidos. Claramente observa-se que os melhores resultados são obtidos com $pert = 5$. Também observa-se que a qualidade dos resultados do algoritmo aumenta quando os valores de $Nreinicios$ e $NIter_{ILS}$ são incrementados. A configuração 10-100-5 do algoritmo ILS_RVND_1 apresentou a melhor média de RPD. Portanto, $Nreinicios = 10$, $NIter_{ILS} = 100$ e $pert = 5$ serão utilizados nos próximos experimentos. Ressalta-se que foram testados valores maiores para $Nreinicios$ e $NIter_{ILS}$. Os resultados não melhoram significantemente, no entanto os tempos de CPU do algoritmo aumentaram significantemente.



(a) Calibração do ILS_RVND_1 (b) Calibração do ILS_RVND_2

Figura 3: Gráfico de médias e intervalos HSD de Tukey com nível de confiança de 95% obtidos na calibração dos algoritmos.

O algoritmo ILS_RVND_2 também foi calibrando utilizando os mesmos valores para os parâmetros $Nreinicios$, $NIter_{ILS}$ e $pert$. Na Figura 3(b) mostra-se o resultado da calibração deste algoritmo. A configuração 10-100-2 do algoritmo ILS_RVND_2 apresentou a melhor média de RPD. Portanto, $Nreinicios = 10$, $NIter_{ILS} = 100$ e $pert = 2$ serão utilizados nos próximos experimentos.

4.3. Resultados para Instâncias Pequenas

No primeiro experimento, os algoritmos ILS_RVND_1, ILS_RVND_1 e GA_LS são comparados com o solver comercial IBM-ILOG CPLEX 12.8 em instâncias de pequeno porte ($N \in \{8, 10, 15, 20\}$, $K \in \{3, 4, 5, 6\}$). O CPLEX é aplicado para resolver o modelo PLIM do problema com um tempo limite de CPU de 3600 segundos para cada instância, e considera-se a melhor solução encontrada nesse tempo. Para todos os métodos (heurísticas e CPLEX) determina-se o desvio percentual relativo (RPD) com relação à melhor solução conhecida (obtida entre todos os métodos). Como os algoritmos heurísticos são estocásticos, cada instância é resolvida 10 vezes por cada algoritmo e consideram-se o melhor resultado e o resultado médio (*Avg.*). Então, para cada algoritmo, dois valores de RPD são calculados.

Na Tabela 1 são apresentados os resultados dos métodos comparados obtidos para as instâncias de pequeno porte. Os resultados para as instâncias com $N = 8$ não são apresentados, pois o CPLEX e as heurísticas determinaram todas as soluções ótimas. Pode-se observar que, para todos os grupos de instâncias, o algoritmo ILS_RVND_1 determina as melhores soluções (apresenta os menores RPDs, *Best* e *Avg*). Também observa-se que o desempenho da GA_LS é melhor que o ILS_RVND_2. Para todas as instâncias, as três heurísticas determinaram soluções melhores que as soluções do CPLEX. No tempo limite, o CPLEX não encontrou nenhuma solução ótima.

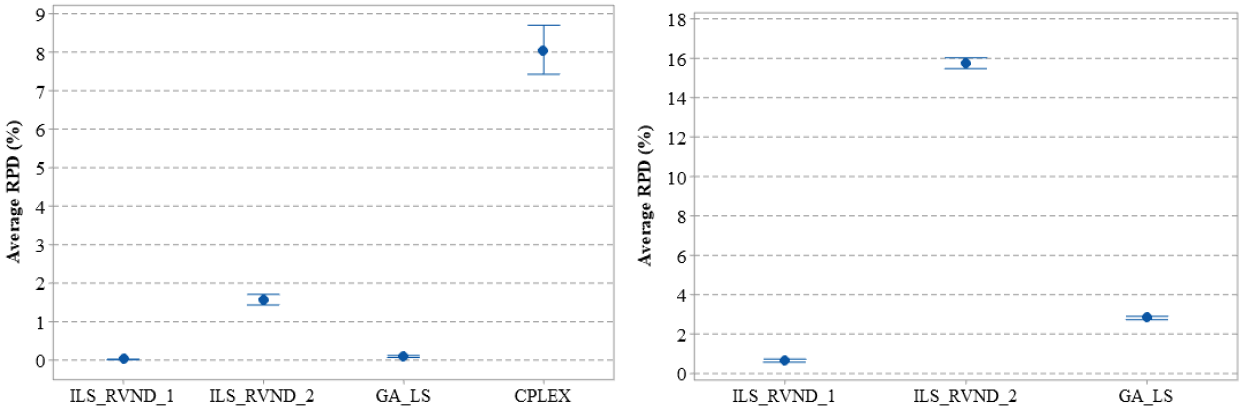
Na Tabela 1 também são apresentados os tempos médios (*Time*) gastos pelos algoritmos e pelo CPLEX. Observa-se que o CPLEX, em todos os grupos de instâncias, gastou o tempo máximo (aproximadamente de 3600 segundos). Para as instâncias maiores ($N = 20$), todas as heurísticas gastaram menos de 2 segundos.

Para validar os resultados obtidos, é feito uma análise estatística usando o RPD como variável de resposta. Para as heurísticas, consideram-se os resultados médios (*Avg*). Esta análise pode ser exibida na Figura 4(a), que mostra o gráfico de médias e os intervalos de Tukey HSD com 95% de nível de confiança. Observa-se que ILS_RVND_1 e GA_LS apresentam as melhores médias

Tabela 1: RPDs médios (%) e tempos de CPU (em segundos) para instâncias de pequeno porte.

	ILS_RVND_1			ILS_RVND_2			GA_LS			CPLEX	
N x *K*	*Best*	*Avg.*	*Time*	*Best*	*Avg.*	*Time*	*Best*	*Avg.*	*Time*	*Best*	*Time*
10 x 3	0,00	0,01	0,11	0,00	0,04	0,16	0,00	0,00	0,17	0,45	3575,10
10 x 4	0,00	0,00	0,14	0,02	0,04	0,25	0,00	0,00	0,22	0,36	3620,62
10 x 5	0,00	0,00	0,17	0,00	0,05	0,37	0,00	0,00	0,27	0,38	3629,82
10 x 6	0,00	0,00	0,20	0,01	0,05	0,45	0,00	0,00	0,33	0,35	3634,85
15 x 3	0,00	0,00	0,33	0,70	1,96	0,30	0,00	0,05	0,53	5,98	3614,83
15 x 4	0,00	0,01	0,39	0,21	0,77	0,46	0,00	0,04	0,62	5,07	3632,21
15 x 5	0,00	0,02	0,47	0,41	0,90	0,71	0,03	0,06	0,74	3,64	3647,36
15 x 6	0,00	0,00	0,54	0,27	0,60	0,91	0,00	0,03	0,86	3,75	3649,92
20 x 3	0,00	0,06	0,79	3,76	5,86	0,51	0,07	0,29	1,25	21,28	3612,73
20 x 4	0,00	0,05	0,83	2,68	4,50	0,76	0,09	0,35	1,34	21,28	3615,70
20 x 5	0,00	0,01	0,99	1,14	2,21	1,11	0,00	0,12	1,61	18,53	3618,98
20 x 6	0,03	0,11	1,09	1,05	1,92	1,52	0,01	0,24	1,74	15,63	3621,38
Média:	0,00	0,02	0,51	0,86	1,58	0,63	0,02	0,10	0,81	8,06	3622,79

(eles são estatisticamente equivalentes) e são significativamente melhor que o ILS_RVND_2. Também pode ser visto que, todos as heurísticas melhoram significativamente os resultados do CPLEX.



(a) Instâncias de pequeno porte.

(b) Instâncias de medio-grande porte.

Figura 4: Gráfico de médias e intervalos HSD de Tukey com nível de confiança de 95% obtidos na comparação dos algoritmos.

4.4. Resultados para Instâncias de Médio-Grande Porte

Para as instâncias de médio-grande porte, os resultados das heurísticas são comparados entre si. Cada instância foi resolvida dez vezes pelas heurísticas e foram considerados o melhor resultado (*Best*) e o resultado médio (*Avg*). A qualidade das soluções encontradas é também avaliada pelo RPD(%). Os RPDs obtidos para as heurísticas são apresentados na Tabela 2. Estes RPDs médios são agrupados pelo número de tarefas (N) e número de veículos (K). Para cada grupo, existem 45 instâncias. Pode-se observar que, para todos os grupos de instâncias, os RPDs *Best* e *Avg* do algoritmo ILS_RVND_1 são menores que os respectivos valores dos outros algoritmos (ou seja, o ILS_RVND_1 supera os outros algoritmos). Considerando a média geral dos algoritmos, o

Tabela 2: RPDs médios (%) e tempos de CPU (em segundos) para instâncias de médio-grande porte.

	ILS_RVND_1			ILS_RVND_2			GA_LS		
N x *K*	*Best*	*Avg.*	*Time*	*Best*	*Avg.*	*Time*	*Best*	*Avg.*	*Time*
50 x 5	0,41	0,89	17,57	12,02	13,91	8,48	0,21	1,48	35,41
50 x 8	0,00	0,41	16,93	10,90	13,14	11,90	0,93	2,22	36,60
50 x 10	0,00	0,31	16,32	10,11	11,86	14,44	1,15	2,38	36,55
50 x 12	0,00	0,31	16,46	9,05	10,75	17,22	1,40	2,43	38,87
80 x 5	0,71	1,63	68,03	14,52	15,80	28,37	0,89	2,40	180,42
80 x 8	0,05	0,65	58,88	16,53	18,05	35,36	1,76	3,38	162,78
80 x 10	0,00	0,59	55,00	16,13	17,63	40,32	2,19	3,43	154,89
80 x 12	0,00	0,40	53,02	14,56	16,19	46,44	2,11	3,29	160,59
100 x 5	0,61	1,24	136,08	14,63	15,95	51,34	0,91	2,44	415,69
100 x 8	0,00	0,69	114,50	17,12	18,90	61,28	2,03	3,40	357,94
100 x 10	0,01	0,46	103,80	16,86	18,56	68,83	2,24	3,57	340,52
100 x 12	0,00	0,45	97,80	16,64	18,17	77,04	2,37	3,48	343,74
Média:	0,15	0,67	62,87	14,09	15,74	38,42	1,52	2,83	188,67

GA_LS é melhor que o ILS_RVND_2. Os algoritmos ILS_RVND_1, ILS_RVND_2 e GA_LS têm médias gerais de 0,67%, 15,74%, e 2,83%, respectivamente. Na Tabela 2 também são apresentados os tempos médios (*Time*) de CPU gastos pelos algoritmos. Observa-se que o algoritmo GA_LS apresenta os maiores tempos e o algoritmo ILS_RVND_2 os menores. Para os melhores algoritmos ILS_RVND_1 e GA_LS, geralmente, o tempo de CPU incrementa quando o número de veículos diminui (ou seja, quando o tamanho das rotas dos veículos é maior).

Considerando os resultados médios (*Avg*), é feito uma análise estatística para verificar se as diferenças observadas nos valores de RPD são de fato significativas. A Figura 4(b) mostra o gráfico de médias e os intervalos de Tukey HSD, com 95% de confiança, obtidos a partir da análise estatística em todas as instâncias de médio-grande porte. Mais uma vez, o melhor desempenho do algoritmo ILS_RVND_1 é claro. Seu intervalo de confiança não se sobrepõe aos intervalos dos outros algoritmos. Isso significa que o ILS_RVND_1 é estatisticamente melhor do que os outros algoritmos e por uma margem significativa. Observa-se também que o segundo melhor algoritmo é o GA_LS.

O ILS_RVND_1 é bem melhor que o ILS_RVND_2, pois este primeiro algoritmo utiliza as sete vizinhanças (Inter e Intra-Rota) que são escolhidas de forma aleatória na busca local RVND_1. No algoritmo ILS_RVND_2, as vizinhanças Intra-Rota somente são utilizadas se houver melhoria da solução com uma vizinhança Inter-Rota. O ILS_RVND_2, frequentemente, utiliza somente as três vizinhança Inter-Rota. Isto torna a busca local RVND_2 muito mais rápido, no entanto perde-se na qualidade da solução.

5. Conclusões

Neste trabalho foi abordado um problema integrado de programação de tarefas numa máquina e o roteamento de veículos com frota heterogênea com o objetivo de minimizar a soma do atraso total ponderado das tarefas, o custo total dos veículos utilizados e o tempo total das viagens. Esse tipo de problema pode ser comumente encontrado em empresas que não mantêm estoque de produtos acabados, realizam uma produção sob encomenda e serviços de entrega expressa. Para o problema foi desenvolvido um modelo matemático de programação linear inteira mista. Por se

tratar de um problema NP-difícil, propõe-se duas heurísticas baseadas nas meta-heurísticas Iterated Local Search (ILS) e Random Variable Neighborhood Descent (RVND). O modelo matemático foi resolvido pelo solver CPLEX, os parâmetros das heurísticas foram calibrados. Experimentos computacionais demonstraram que algoritmo ILS_RVND_1 apresenta melhores soluções que o algoritmo ILS_RVND_2 e o algoritmo AG_LS da literatura.

Referências

Archetti, C., Feillet, D., e Speranza, M. G. (2015). Complexity of routing problems with release dates. *European journal of operational research*, 247(3):797–803.

Billaut, J.-C., Della Croce, F., e Ta, Q. C. (2017). Tabu search and matheuristic algorithms for solving an integrated flow shop and vehicle routing problem. In *12th Metaheuristics International Conference (MIC)*.

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

Chang, Y.-C., Li, V. C., e Chiang, C.-J. (2014). An ant colony optimization heuristic for an integrated production and distribution scheduling problem. *Engineering Optimization*, 46(4):503–520.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: review and extensions. *Operations research*, 58(1):130–148.

Cheng, B.-Y., Leung, J. Y.-T., e Li, K. (2015). Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method. *Computers & Industrial Engineering*, 83:217–225.

Condotta, A., Knust, S., Meier, D., e Shakhlevich, N. V. (2013). Tabu search and lower bounds for a combined production–transportation problem. *Computers & operations research*, 40(3): 886–900.

Du, J. e Leung, J. Y.-T. (1990). Minimizing total tardiness on one machine is np-hard. *Mathematics of operations research*, 15(3):483–495.

Ghannadpour, S. F. e Zarrabi, A. (2019). Multi-objective heterogeneous vehicle routing and scheduling problem with energy minimizing. *Swarm and evolutionary computation*, 44:728–747.

Hall, N. G. e Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, 51(4):566–584.

Hansen, P. e Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467.

Karaoğlan, İ. e Kesen, S. E. (2017). The coordinated production and transportation scheduling problem with a time-sensitive product: a branch-and-cut algorithm. *International Journal of Production Research*, 55(2):536–557.

Liu, L., Li, W., Li, K., e Zou, X. (2020). A coordinated production and transportation scheduling problem with minimum sum of order delivery times. *Journal of Heuristics*, 26(1):33–58.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, p. 320–353. Springer.

Molina-Sánchez, L. e González-Neira, E. (2016). Grasp to minimize total weighted tardiness in a permutation flow shop environment. *International Journal of Industrial Engineering Computations*, 7(1):161–176.

Penna, P. H. V., Subramanian, A., e Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232.

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

Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research*, 227(1):152–165.

Wang, J., Yao, S., Sheng, J., e Yang, H. (2019). Minimizing total carbon emissions in an integrated machine scheduling and vehicle routing problem. *Journal of Cleaner Production*, 229:1004–1017.

Xiao, Y. e Konak, A. (2015). A simulating annealing algorithm to solve the green vehicle routing & scheduling problem with hierarchical objectives and weighted tardiness. *Applied Soft Computing*, 34:372–388.

Zarandi, M. H. F., Asl, A. A. S., Sotudian, S., e Castillo, O. (2020). A state of the art review of intelligent scheduling. *Artificial Intelligence Review*, 53(1):501–593.

Zou, X., Liu, L., Li, K., e Li, W. (2018). A coordinated algorithm for integrated production scheduling and vehicle routing problem. *International Journal of Production Research*, 56(15):5005–5024.