Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Solution methods for the integrated permutation flowshop and vehicle routing problem

Marcelo Seido Nagano[a*], Caio Paziani Tomazella[a], Roberto Fernandes Tavares-Neto[b] and Levi Ribeiro de Abreu[a]

[a]*University of São Paulo, Department of Production Engineering, São Carlos, Brazil*
[b]*Federal University of São Carlos, Department of Industrial Engineering, São Carlos, Brazil*

C H R O N I C L E

Article history:
Received: October 15, 2021
Received in revised format: December 25, 2021
Accepted: January 29, 2022
Available online:
January 29, 2022

Keywords:
Integrated Scheduling
Flowshop
Distribution
Mixed-Integer Programming
Iterated Greedy

A B S T R A C T

The integration between production and distribution to minimize total elapsed time is an important issue for industries that produce products with a short lifespan. However, the literature focus on production environments with a single stage. This paper enhances the complexity of the production system of an integrated production and distribution system by considering flowshop environment decisions integrated with a vehicle routing problem decision. In this case, each order is produced in a permutation flowshop subsystem and then shipped to its destination by a capacitated vehicle, and the objective is to sequence these orders to minimize the makespan of the schedule. This paper uses two approaches to address this integrated problem: a mixed-integer formulation and an Iterated Greedy algorithm. The experimentation shows that the Iterated Greedy algorithm yields results with a 0.02% deviation from the optimal for problems with five jobs and is a viable option to be used in practical cases due to its short computational time.

1. Introduction

In recent years, researchers and practitioners have been focusing their efforts to plan decisions across different but integrated production subsystems. There is a considerable body of literature regarding the integration between production and distribution issues (Chen 2004; Moons et al., 2017). There are claims to support the need for this integration such as improved service level, usually related to the minimization of total system flowtime (Ulrich, 2013), and total system cost (Rohmer & Billaut 2015). Moreover, an integrated approach is of great interest and importance to industries that produce perishable goods (Amorim et al., 2013; Viergutz and Knust, 2014; Belo-Filho, Amorim & Almada-Lobo, 2015; Devapriya et al., 2017), chemical components (Canton, 2003; Kergosien et al., 2017) and other products with short lifespan as newspapers (Russell, Chiang et al., 2008). According to the literature, most of the applications presented so far represent manufacturing environments as a single stage production, and a little has been done regarding multi-stage manufacturing environments, such as flowshops and jobshops (Moons et al., 2017). In our research, the flowshop environment was adopted in some articles (Scholz-Reiter et al., 2011; Rohmer & Billaut, 2015; Ehm & Freitag, 2016; Ramezanian et al., 2017). These authors combine production and routing decisions to minimize supply chain related costs, such as production, inventory and tardiness.

* Corresponding author.
E-mail address: drnagano@usp.br (M. S. Nagano)

© 2022 Growing Science Ltd. All rights reserved.
doi: 10.5267/j.jpm.2022.1.002

This article presents a Mixed-Integer Problem (MIP) formulation and a constructive heuristic, both considering the two subsystems simultaneously. The Integrated Production and Distribution Problem (IPDP) studied considers a permutation flowshop manufacturing subsystem connected to a routing distribution subsystem by an unbounded inventory. The distribution is done by a single vehicle with limited capacity, which requires a pre-determined time to move between each destination. A number of clients from different locations are served by the system, and to each one of them a job is assigned. The performance criterion of the problem is the makespan of the whole sequence, which is the time elapsed between the start of the production in the flowshop and the arrival of the distribution vehicle to the origin after its last delivery.

Using the notation from Z.-L. Chen (2010), this problem is written as $F_m \mid \mid V(1, Q), routing \mid n \mid C_{max}$. In this notation, the first term refers to the machine layout, the second to any restrictions of the production subsystem, the third to the number and capacity of vehicles and the delivery method, the fourth to the number of clients and the fifth to the performance criteria. Since both subsystems, the permutation flowshop problem (Pinedo, 2008) and the vehicle routing problem (Lenstra and Kan, 1981) are NP-hard, the integrated problem addressed in this article is also NP-hard.

To solve this problem, defined in Section 2, two approaches are adopted: a MIP is formulated in Section 3; and an Iterated Greedy (IG) algorithm is presented in Section 4. All methods were implemented and the results are shown in Section 5. Afterwards, final remarks are presented in Section 6.

2. Literature Review

Despite recent developments in the field, the literature brings important insights to define an IPDP. Two main groups of IPDPs can be found in the literature, according to different organizational decision levels:

- A lot-sizing problem approach, usually focusing on decisions of "when" and "how much" to produce to improve a cost indicator. These are tactical level decisions, covering longer planning horizons with higher aggregation of products and operations (Maxwell, 1964; Elmaghraby, 1978; Jans & Degraeve, 2008).
- A production subsystem approach, dealing with manufacturing environments such as a single machine and parallel machines (Moons et al., 2017). This approach focuses on operational level decisions of "in which sequence" to produce the orders, and most of its straightforward indicators are time- and cost-related. These decisions consider more operational details and constraints regarding the orders, machines and resources involved.

These two approaches generate entirely different problems, both in mathematical formulations and in solution methods. This article uses the second approach to the IPDP, therefore the literature review is focused on which type of production subsystem is considered, as well as how the distribution is made.

Focused on the production subsystem approach, Moons et al. (2017) presented a detailed review of the state-of-the-art literature on IPDPs, and concluded that most articles consider single-stage production subsystems, either with a single or parallel machines. Regarding the restrictions of these subsystems, such as setup times and release or availability dates, most authors do not consider them in the modelling of problems, despite their impact on the production process.

Still defining an IPDP, the distribution subsystem is usually represented by two patterns: direct shipments to a single location (Ng and Lu, 2012) or a routing-enable distribution (Gao, Qi, and Lei, 2015). According to Moons et al. (2017), the majority of articles use similar vehicles with limited capacity for distribution. The number of vehicles can be pre-determined or not: Devapriya et al. (2017) used in their model an open number of vehicles that can be hired at a cost depending on the necessity, while Cheng et al. (2017) limited the number of vehicles to one.

In more recent articles, a more focused approach on the distribution of perishable goods (Devapriya et al. 2017; Kergosien et al., 2017) and the use of meta and matheuristics to achieve high quality solutions is noticed (Devapriya et al., 2017; Darvish & Coelho 2018). Cheng et al. (2017) and Kergosien et al. (2017) proposed sequential approaches to the problems, solving the production subsystem, followed by the distribution routing. Devapriya et al. (2017) prioritized the distribution decisions, since their problem involved a single product, and Darvish and Coelho (2018) showed the dominance of the integrated approach over the sequential for a supply chain problem with location, production, inventory and direct shipment distribution.

As for IPDPs with a flowshop as the production subsystem, the articles found in the literature show different variations of the flowshop problem being addressed. Rohmer and Billaut (2015) presented methods to solve integrated problems with a permutation flowshop using a sequential approach, solving the production subsystem first and using its results as inputs to obtain the distribution decisions.

Other variations of the flowshop problem found in IPDP articles are: open flowshop (Scholz-Reiter et al., 2011); flexible flowshop (Ehm & Freitag, 2016); and non-permutation flowshop (Ramezanian et al., 2017). In all these articles, the authors approached the problems with the objective to minimize the total operation costs of the supply chain, including inventory (from both work-in-progress and finished goods), transportation and tardiness. Therefore, the proposal of this article is novel in a sense that it considers time-related factors as its objective.

Regarding the distribution systems, the cited authors consider both single and multiple-vehicle fleets, always having the carrying capacity as a constraint. The distribution in the model from Rohmer and Billaut (2015) is done by a third-party service provider, and Ramezanian et al. (2017) addressed two distribution types, direct shipping and vehicle routing. Finally, Ehm and Freitag (2016) confirmed the advantages of using the integrated approach, showing their model gave solutions with up to 30% lower costs than those obtained by solving the production and routing problems separately.

As for the use of the IG algorithm for IPDPs, Tavares-Neto and Nagano (2018) applied it for the problem with multiple parallel machines with sequence dependent setup times and a single capacitated vehicle for distribution ($P_m \mid s_{jl} \mid V(1,Q) \mid n \mid C_{max}$). The authors showed that the use of the algorithm greatly improved the initial solutions obtained through a constructive heuristic and also outperformed the adaptation of a genetic algorithm proposed for a similar problem (Ulrich 2013). Finally, Abreu, Tavares-Neto and Nagano (2021) presented a biased random key genetic algorithm with hybridization of IG for open shop with routing by a capacitated single vehicle. The proposed method got competitive results.

3. A Mixed-Integer Model

This section presents a MIP formulation for the $F_m \mid \mid V(1,Q), routing \mid n \mid C_{max}$ problem. The objective is to minimize the value of variable C_{max}, which corresponds to the makespan of the integrated schedule of the system. The restrictions are divided into four main groups: flowshop, routes, capacity and integration.

The symbols used to define a problem instance are given in Table 1, and a solution is characterized by the decision variables presented in Table 2.

Table 1
Sets and parameters used to define a problem instance

Symbol	Description
j, l	Indexes used to identify processing orders, varying from 0 (dummy job) to n
i	Index used to identify flowshop machines, varying from 1 to m
r	Index used to identify a route, varying from 1 to n
p_{ij}	The processing time of order j on machine i
σ_j	The size (e.g., volume) of job j
ψ	The total capacity of the vehicle in the same unity used for σ_j
δ_{jl}	The time required for the vehicle to travel the distance between j and l, measured in the same unity used for p_{ij}

Table 2
Variables and constants used to define a solution

Variable	Type	Description
x_{lj}	Binary	1 if job l immediately precedes j in the flowshop; 0 otherwise
w_{ljr}	Binary	1 if job j is delivery immediately after l in route r; 0 otherwise
C_{ij}	Integer	Moment when job j is released by machine i
A_j	Integer	Total occupied capacity on the vehicle when j is allocated to a route
R_r	Integer	Release date of route r
D_j	Integer	Delivery date of job j, not including the dummy job
C_{max}	Integer	The makespan of the sequence (when the vehicle returns after the last job is delivered)
GP	Constant	A very large number regarding the production system
G_σ	Constant	A very large number regarding the vehicle capacity constrains
M	Constant	A very large number regarding the distribution system

3.1 Flowshop Modelling

The first group of restrictions models the functioning of a permutation flowshop. Eq. (1) calculates a Big M for the processing times. Constraints (2) and (3) guarantee that exactly one job is allocated to each position of the sequence. Constraints (4) set the dummy job completion times to zero, Constraints (5) guarantee that the machines follow the same order of jobs, and Constraints (6) guarantee that machine i will only start processing a job after it is released from $i-1$.

$$GP = \sum_i \sum_j p_{ij} \tag{1}$$

$$\sum_{l=1}^{n} x_{lj} = 1 \quad \forall \quad 0 \le j \le n \tag{2}$$

$$\sum_{j=1}^{n} x_{lj} = 1 \quad \forall \quad 0 \le l \le n \tag{3}$$

$$C_{i0} = 0 \quad \forall \quad 1 \le i \le m \tag{4}$$

$$C_{ij} \ge C_{il} + p_{ij} + (x_{lj} - 1)GP \quad \forall \begin{cases} 1 \le i \le m \\ 0 \le l \le n \\ 1 \le j \le n \end{cases} \tag{5}$$

$$C_{ij} \ge C_{(i-1)j} + p_{ij} \quad \forall \begin{cases} 2 \le i \le m \\ 0 \le j \le n \end{cases} \tag{6}$$

3.2 *Routes Modelling*

This group of constraints is responsible to model the routes which the vehicle follows to distribute the finished jobs among the clients. Since the number of needed routes is initially unknown, the upper bound of n routes (one for each job) is used in order to generate the constraint groups. The constraints are programmed in such a way that the routes are only activated when needed, while the inactive have all w_{ljr} values set to zero so they are not computed in the fitness function. Constraints (7) guarantee that each location is visited exactly once. Constraints (8) and (9) force the routes to start and end at the origin, Constraints (10) maintain the routes flow along the destinations and Constraints (11) do not allow the vehicle to have the same client as origin and destination. Finally, Constraints (12) allocate the empty routes to the end of the schedule.

$$\sum_r \sum_l w_{ljr} = 1 \quad \forall \quad 1 \le j \le n \tag{7}$$

$$\sum_j w_{0jr} = 1 \quad \forall \quad 1 \le r \le n \tag{8}$$

$$\sum_j w_{j0r} = 1 \quad \forall \quad 1 \le r \le n \tag{9}$$

$$\sum_{l \ne j} w_{ljr} = \sum_{l \ne j} w_{jlr} \quad \forall \begin{cases} 0 \le j \le n \\ 1 \le r \le n \end{cases} \tag{10}$$

$$w_{jjr} = 0 \quad \forall \begin{cases} 1 \le j \le n \\ 1 \le r \le n \end{cases} \tag{11}$$

$$w_{00(r-1)} \le w_{00r} \quad \forall \quad 1 \le r \le n \tag{12}$$

3.3 *Vehicle Capacity Modelling*

The goal of this restriction group is to guarantee that the vehicle is not loaded over its capacity. Eq. (13) calculates a Big M for the vehicle load. Constraint (14) the load of an empty vehicle to zero. Constraints (15) calculate, considering that the jobs are loaded into the vehicle in the same order that they are delivered, the capacity of the vehicle that is occupied when each job is added to a route. Constraints (16) guarantee that the total capacity does not exceed Ψ.

$$G_\sigma = \sum_j \sigma_j \tag{13}$$

$$A_0 = 0 \tag{14}$$

$$A_j \geq A_l + \sigma_j - (1 - w_{ljr})G_\sigma \quad \forall \begin{cases} 1 \leq j \leq n \\ 0 \leq l \leq n, l \neq j \\ 1 \leq r \leq n \end{cases} \tag{15}$$

$$A_j \leq \Psi \quad \forall \quad 1 \leq j \leq n \tag{16}$$

3.4 Production and Distribution Integration Modelling

The fourth and last restriction group integrated both systems, defining the release and arrival times of each route and the delivery times of each job. Constraints ([r1]) and ([r2]) guarantee, respectively, that the vehicle departs for route r after all its jobs are finished in the flowshop and after it arrives from the previous route. Constraints ([d1]) and ([d2]) calculate the arrival of each job according to its designated position on a route.

$$R_r \geq C_{mj} - (1 - w_{ljr})GP \quad \forall \begin{cases} 1 \leq r \leq n \\ 1 \leq j \leq n \\ 0 \leq l \leq n \end{cases} \tag{17}$$

$$R_r \geq D_l + \delta_{l0} - (1 - w_{l0(r-1)})GP \quad \forall \begin{cases} 1 \leq l \leq n \\ 2 \leq r \leq n \end{cases} \tag{18}$$

$$D_j \geq R_r + \delta_{0j} - (1 - w_{0jr})GP \quad \forall \begin{cases} 1 \leq j \leq n \\ 1 \leq r \leq n \end{cases} \tag{19}$$

$$D_j \geq D_l + \delta_{lj} - (1 - \sum_r w_{ljr})GP \quad \forall \begin{cases} 1 \leq l \leq n \\ 0 \leq j \leq n, j \neq l \end{cases} \tag{20}$$

3.5 Makespan Function

The problem aims at minimizing C_{max}, which is done by using Constraints (22). Eq. (21) sets a Big M for the makespan by summing up all processing times and distances. The value of C_{max} is than set to be the latest instant when the vehicle returns from a route. Constraints (22) are activated whenever $w_{j0r} = 1$, meaning that j is the last delivered job of route r; than the instant when the vehicle arrives back at the origin is calculated by summing the delivery date of j (D_j) plus the time needed for the vehicle to return (δ_{j0}). The total makespan is than obtained by taking the largest of these values, which is from the last activated route.

$$M = \sum_i \sum_j p_{ij} + \sum_l \sum_j \delta_{lj} \tag{21}$$

$$C_{max} \geq D_j + \delta_{j0} - (1 - w_{j0r})M \quad \forall \begin{cases} 0 \leq j \leq n \\ 1 \leq r \leq n \end{cases} \tag{22}$$

In this article, the proposed method of using a software to solve the MIP formulation is addressed as MIP_{TTN}.

4. An Iterated Greedy Algorithm

In this section we present an IG algorithm for the $F_m \mid \mid V(1, Q), routing \mid n \mid C_{max}$ problem, called IG_{TTN}. IG_{TTN} uses a constructive heuristic in order to obtain an initial solution, which is shown in Subsection 4.1.

4.1 Initial Solution and Insertion Mechanism

The algorithm used to generate an initial solution for the IG is an adaptation of the I/O heuristic from Tavares-Neto and Nagano (2018) for this problem. The I/O algorithm itself is an extension of the NEH heuristic (Nawaz, Enscore, and Ham 1983) to integrate parallel-machine scheduling with the distribution subsystem. It sets the jobs in an initial sequence and applies an insertion mechanism to sequence the jobs for both production and distribution. This heuristic, named I/O_{TTN} for future reference, uses a similar mechanism, with the difference in being applied to a flowshop production subsystem instead of a parallel machines. I/O_{TTN} consists of two main phases, and its pseudo-code is presented in Figure 1:

- **Phase 1 - Initial Sequencing (line 1)**: The jobs are ordered to form an initial solution for the next phase. In this article, five different orderings were tested for the proposed algorithm.
- **Phase 2 - Insertion Mechanism (lines 5-19)**: The heuristic solution is constructed by inserting each job following the initial sequence. The solution is initiated containing the first job, then, one by one, the following jobs are tested in each available position, and the one which gives the smaller fitness value is kept to the next insertion.

The extension proposed for integrating the flowshop scheduling with distribution consists in applying an insertion mechanism on the routes: for each position tested in the insertion phase, the job is inserted in all possible positions within existing routes (lines 11-15) and in a new route (line 16). Thus, this algorithm produces two sequences: a sequence S_π related to the production schedule and a sequence V_π related to the distribution schedule.

With a fixed production schedule, the distribution problem can be approached as a single-machine production problem with release dates, in which each job is a route and the release date for each route is the time when all jobs from it are finished in the production stage. Since the objective is to minimize the makespan, the routes are then sequenced in non-decreasing order of release dates, and the makespan is calculated. Note that for each different job insertion the release dates must be updated.

```
$J \leftarrow$ set of jobs in an initial order;
$S_\pi, S^*_\pi \leftarrow$ current and best flowshop production sequences;
$V_\pi, V^*_\pi \leftarrow$ current and best delivery sequences;
$S_\pi = S^*_\pi = V_\pi = V^*_\pi = 0$;
foreach $j \in J$ do
    $Fitness^* = \infty$;
    for $ps = 1...|S_\pi|$ do
        $S'_\pi = S_\pi$;
        Insert j in the ps-th position of $S'_\pi$;
        Calculate completion times of jobs in $S'_\pi$;
        foreach existing rout r do
            Insert j in the position that adds minimum distance covered by r;
            Re-order the routes by release date;
            Update values of $Fitness*$, $S^*_\pi$ and $V^*_\pi$ with the best found so far;
        end
        Insert j in a new route and place it in the position given by its release date;
        Update values of $Fitness*$, $S^*_\pi$ and $V^*_\pi$ if necessary;
    end
end
Return $S^*_\pi, V^*_\pi$;
```

Fig. 1. The pseudo-code for the I/O_{TTN} heuristic

4.2 The $\boldsymbol{IG_{TTN}}$ Algorithm

IG_{TTN} is a direct extension of I/O_{TTN}, allowing better solutions to be found by using local search techniques, as shown by Ruiz and Stützle (2007). As previously mentioned, Tavares-Neto and Nagano (2018) successfully applied this approach to $P_m \mid s_{jl} \mid V(1,Q),\ routing \mid n \mid C_{max}$, showing the efficiency of this method over an evolutionary algorithm.

IG_{TTN} is divided in three steps, and its pseudo-code is shown in Fig. 2:

- **Initial Solution (line 1)**: I/O_{TTN} is applied once to construct a initial solution;
- **Destruction Phase (line 6)**: a fixed number of jobs is removed from the current solution;
- **Reconstruction Phase (lines 7-11)**: the removed jobs are reinserted in the solution using the same insertion mechanism from I/O_{TTN};

The Destruction and Insertion Phases are repeated in sequence ig_{cyc} times, allowing the algorithm to perform the improvements on different subsections of the current solution. In each iteration, the number of removed jobs is $ig_{jobs} = n \times ig_{perc}$, in which ig_{perc} is a percentage of the total of jobs in the problem. Section 5 shows how both ig_{cyc} and ig_{perc}

values are obtained. Whenever the solution found is better than the current one, it is updated with the former and used as the starting point in the following cycle. Both number of cycles and number of jobs to be removed are fixed parameters of the algorithm.

```
$P_\pi, P_\pi^*$ ← current and best IPDP solution, obtained by $I/O_{TTN}$;
$ij_{cyc}$ ← number of cycles of the IG algorithm;
$ij_{jobs}$ ← number of jobs removed in each iteration;
Fitness* ← Fitness ( $P_\pi^*$ );
foreach $0<j<ig_{cyc}$ do
    Remove $ig_{jobs}$ randomly picked jobs from $P_\pi$ ;
    Reinsert the jobs back in $P_\pi$ using the insertion mechanism from Algorithm 1;
    if Fitness ( $P_\pi$ ) < Fiteness* then
        Fiteness* = Fitness ( $P_\pi^*$ );
        $P_\pi^* = P_\pi$
    end

end
return $P_\pi^*$ ;
```

Fig. 2. The pseudo-code for the IG_{TTN} algorithm.

5. Results and Analysis

For performance testing, a set of randomly generated instances was created. Since this problem was not previously studied in the literature, no benchmark databases are available. The parameters for instance generation were taken from Tavares-Neto and Nagano (2018), with the exception of the number of machines in the flowshop subsystem, for which the range from the benchmark of Taillard (1993) was used.

Each instance is defined by four main parameters, $n = \{5,10,20,40,80\}$, $m = \{5,10,15,20\}$, $\theta_d = \{10,20,30\}$ and $\theta_m = \{10,20,30\}$ (θ_d and θ_m are both used to define the plane in which the destinations are located). A total of 180 combinations of these parameters are possible. With 50 unique instances generated for each, the database consists of 9000 problem instances.

The constant values in the instances were randomly and uniformly distributed in the following ranges: $p_{ij} = [1,99]$; $\sigma_j = [1,10]$; $\psi = [\max\sigma_j, 5 \times \max\sigma_j]$. The positions of the delivery locations were also randomly and uniformly distributed in a $\theta_d \times \theta_d$ Cartesian place. The values of δ_{jl} were obtained by multiplying the Euclidian distance between j and l by θ_m.

In order to define the parameters of the IG algorithm, the IRACE package (López-Ibáñez et al. 2016) was used. The initial set of parameters was: number of cycles $ig_{cyc} = \{50,100,150, 200,250,300\}$; percentage of jobs removed from the sequence in the destruction phase $ig_{perc} = \{20,40,60,80\}$. These values were based on the parametrization performed by Tavares-Neto and Nagano (2018) and allow a total of 24 different parameter combinations. The results obtained from the IRACE parametrization were the following: $ig_{cyc} = 300$ and $ig_{perc} = 40$.

Five different orders were tested for the initial sequence. The objective of considering all possibilities is to analyse the impact of the initial sequence in the final solution according to the parameters of the problem instances.

- **SPT** Sequences the jobs in non-decreasing total processing times order;
- **LPT** Sequences the jobs in non-increasing total processing times order;
- **NNDIST** Sequences the jobs by applying the Nearest Neighbour Search from the origin;
- **SIZE** Sequences the jobs in non-decreasing sizes order;
- **SIZEDEC** Sequences the jobs in non-increasing sizes order;

MIP_{TTN} was modelled and solved using Python 3.5.0 and IBM ILOG CPLEX 12.7.0, while IG_{TTN} was coded in C++ using Microsoft Visual Studio 2017. Both experimentations were performed in an Intel Core i5-2500K CPU running at 3.30GHz, with 4GB RAM and Windows 7 Professional.

Table 3 shows the results of MIP_{TTN} according to the output. Due to the complexity of the formulation, it was only able to solve optimally the set of problems with 5 jobs within the time limit (3.600 seconds). For the problems with $n = 10$ and almost half of the $n = 20$ set, a feasible solution was found. No feasible solution was reached for the remainder of problems, and none of those with $n = 80$ could be compiled.

Table 3
Results of $\boldsymbol{MIP_{TTN}}$ within 1 hour of processing time

Status	**Total of problems**
Optimal solution found	1800
Time limit exceeded; integer solution found	2668
Time limit exceeded; no integer solution	2732
The compilation was not possible	1800

5.1 Results for instances with n = 5

The first analysis compares the results from IG_{TTN} to those obtained by MIP_{TTN}, considering only the problems with optimal solutions found ($n = 5$). From the 1800 instances, IG_{TTN} found the optimal solution in 91% of them, with little variation regarding the initial order. As for the overall quality of the solutions, the best performing algorithms used the LPT and $NNDIST$ sequences, showing a 0.12% Average Relative Deviation (ARD) to the optimal values. The values for all initial sequences are presented in Table 4.

Table 4
$\boldsymbol{IG_{TTN}}$ results for $\boldsymbol{n = 5}$, sorted by initial sequence

Sequence	**Optimal solutions found**	$\boldsymbol{ARD}$
SPT	1653	0.13%
LPT	1647	0.12%
$NNDIST$	1645	0.12%
$SIZE$	1633	0.15%
$SIZEDEC$	1636	0.15%

Table 5 shows the average computational time in ms for each method to solve the problem instances, grouped by the number of machines. While the relative difference between the MIP_{TTN} and IG_{TTN} is big, the former is still a viable method to be used for small size instances in practical cases. For the more complex instances ($m = 20$) MIP_{TTN} takes an average of $4.18s$ to find the optimal solution, and on the worst case of all the 1800 problems, $37.14s$.

Table 5
Average computational time (in $\boldsymbol{ms}$) for each method to solve the small instances

$\boldsymbol{m}$	$\boldsymbol{IG_{TTN}}$	$\boldsymbol{MIP_{TTN}}$
5	1.6	1566.3
10	2.5	2368.4
15	3.5	3119.1
20	4.5	4179.5

5.2 Results for all instances

In this section, the results of MIP_{TTN} are not considered since they were obtained for problems with $n = 5$ only. For each instance, the minimum value obtained among the five different IG_{TTN} possibilities is taken as the best solution and the ARD is calculated relative to it. Tables 6 and 7 sort the IG_{TTN} ARD by the instances size (n and m) and distance parameters (θ_d and θ_m) respectively.

From Table 6, it is possible to observe that for small and medium sized instances ($n \leq 20$), the initial sequence has little to no effect in the performance of IG_{TTN}. However, for large instances ($n \geq 40$), $NNDIST$ consistently provides the best initial solution for the algorithm. A similar trend is observed in Table 7, in which $NNDIST$ provides the best solution for problems with $\theta_d \geq 20$ (the origin and destinations are placed within wider areas).

A Tukey's HSD test was used in order to verify the dominance of $NNDIST$ to the remaining orders. The test was done in the PASW Statistics 17.0 software, using the 9000 ARD values obtained for each initial sequence. The results are presented in Table 8, and show that $NNDIST$ is statistically superior to the others with 95% confidence. According to Columns 1 and 2, the ARD obtained with $NNDIST$ is significantly lower than those obtained with the other sequences, while these other four have no significant difference among them.

Table 6
ARD of IG_{TTN} for each initial sequence, when compared to the minimum value found. Results sorted by n and m

n	m	SPT	LPT	$NNDIST$	$SIZE$	$SIZEDEC$
5	5	0.12%	0.12%	**0.10%**	0.18%	0.14%
	10	0.16%	0.12%	0.14%	**0.09%**	0.17%
	15	0.09%	**0.08%**	0.12%	0.18%	0.14%
	20	**0.07%**	0.09%	0.06%	0.10%	0.08%
10	5	0.55%	0.41%	0.42%	**0.38%**	0.45%
	10	0.42%	**0.37%**	0.54%	0.41%	0.42%
	15	0.42%	**0.28%**	0.31%	0.31%	0.41%
	20	**0.29%**	0.30%	0.32%	0.34%	0.35%
20	5	1.72%	**1.34%**	1.43%	1.47%	1.56%
	10	1.33%	1.34%	1.31%	**1.18%**	1.32%
	15	1.42%	1.35%	1.42%	**1.32%**	1.36%
	20	**1.10%**	1.16%	1.18%	1.13%	1.21%
40	5	1.97%	2.20%	**1.78%**	2.16%	1.97%
	10	1.97%	1.93%	**1.83%**	2.10%	1.93%
	15	1.88%	1.96%	**1.78%**	1.99%	1.80%
	20	1.74%	1.87%	**1.62%**	1.81%	1.78%
80	5	1.92%	2.11%	**1.57%**	2.01%	1.86%
	10	1.86%	1.88%	**1.46%**	1.85%	1.81%
	15	1.78%	1.81%	**1.43%**	1.75%	1.70%
	20	1.68%	1.83%	**1.52%**	1.74%	1.56%

Table 7
ARD of IG_{TTN} for each initial sequence, when compared to the minimum value found. Results sorted by θ_d and θ_m

θ_d	θ_m	SPT	LPT	$NNDIST$	$SIZE$	$SIZEDEC$
10	10	0.45%	0.45%	**0.42%**	0.43%	0.45%
	20	1.06%	1.08%	**0.93%**	1.08%	1.01%
	30	1.27%	1.28%	1.26%	1.27%	**1.13%**
20	10	1.05%	1.04%	**0.91%**	1.06%	1.03%
	20	1.28%	1.26%	**1.20%**	1.27%	1.26%
	30	1.24%	1.29%	**1.08%**	1.24%	1.26%
30	10	1.25%	1.24%	**1.13%**	1.24%	1.25%
	20	1.22%	1.23%	**1.04%**	1.24%	1.20%
	30	1.32%	1.27%	**1.19%**	1.29%	1.32%

Table 8
Tukey's HSD test results for the initial sequences

Sequence	Subset for $\alpha = 0.05$	
	1	2
$NNDIST$	0.0102	
$SIZEDEC$		0.0110
SPT		0.0112
$SIZE$		0.0112
LPT		0.0113

Some observations can be made when comparing IG_{TTN} to the IG algorithm from Tavares-Neto and Nagano (2018). Aside from the production subsystem, the remaining parts of the problem are similar. In both methods, the parametrization results set the same number of jobs removed in each cycle (ig_{jobs}) in 40% of the total.

Regarding the initial sequence, Tavares Neto (2016) showed that the most efficient order was obtained by applying the Nearest Neighbour Search on the jobs sequence dependent setup times. This sequencing was not possible in this article since the $F_m||V(1,Q), routing|n|C_{max}$ problem does not contain setup times, therefore the $NNDIST$ is the dominant initial order. These results indicate that sequences that consider the relationships between jobs more effective than sequencing based on (non-)increasing values.

Having shown the quality of the solutions obtained with IG_{TTN}, a concern related to its practical use arises from the time required for its execution, given its high complexity. However, as can be seen on Table 9, even for the largest of instances ($n = 80, m = 20$) the computational time of IG_{TTN} averages 12.63 seconds, which shows that this algorithm is a viable option to be used in real-life cases, providing high quality, near optimal solutions in feasible time.

Table 9
Computational time (in s) required for IG_{TTN} execution

n	m			
	5	10	15	20
5	0.00	0.00	0.00	0.00
10	0.01	0.02	0.02	0.03
20	0.06	0.10	0.15	0.20
40	0.46	0.80	1.16	1.53
80	3.89	6.79	9.68	12.68

6. Final Remarks

This article studied the IPDP consisting of permutation flowshop and delivery to multiple destinations done by a single vehicle with limited capacity. A MIP formulation and an Iterated Greedy algorithm are proposed in order to solve a set of 9000 randomly generated instances.

Due to computational constraints, the optimal solutions obtained through the MIP formulation are limited to problems with 5 jobs only. For these problems, IG_{TTN} yields results within a 0.12% range from the optimal solution, while finding the best value in 9 out of 10 problems. Regarding bigger problems, to which the optimal solutions are not available, the analysis shows the viability of using IG_{TNN} due to its short computational time (13 seconds in the most complex case) and suggests the use of the Nearest Neighbour search on distance values in order to generate an initial solution for the algorithm.

Overall, the presented methods are highly adaptable for different problems, indicating that they can be applied to more complex scenarios and used in real-world industrial cases.

As guidelines for future research, the following suggestions are made: study possibilities of improvements in the destruction and reconstruction phases of IG_{TNN} in order to improve the final results and reach higher percentages of optimal solutions found for small-size instances; apply the proposed Iterated Greedy algorithm and the MIP formulation for problems with additional features such as machine set-up times and vehicle loading/delivery times; analyze the effect of using of multiple vehicles instead of a single one in the makespan of the schedule; analyze this problem and the proposed methods while taking into consideration cost-related performance measures.

Acknowledgement

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) [grant numbers 306075/2017-2, 430137/2018-4 and 312585/2021-7].

References

Abreu, L. R., Tavares-Neto, R. F., & Nagano, M. S. (2021). A new efficient biased random key genetic algorithm for open shop scheduling with routing by capacitated single vehicle and makespan minimization. *Engineering Applications of Artificial Intelligence*, *104*, 104373.

Amorim, P., Belo-Filho, M. A. F., Toledo, F. D., Almeder, C., & Almada-Lobo, B. (2013). Lot sizing versus batching in the production and distribution planning of perishable goods. *International Journal of Production Economics*, *146*(1), 208-218.

Belo-Filho, M. A. F., Amorim, P., & Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research*, *53*(20), 6040-6058. https://doi.org/10.1080/00207543.2015.1010744.

Canton, J. (2003). Integrated Support System for Planning and Scheduling of Batch Chemical Plants. PhD thesis, Universitat Politecnia de Catalunya.

Chen, Z.-L. (2004). Integrated Production and Distribution Operations: Taxonomy, Models, and Review. In *Handbook of Quantitative Supply Chain Analysis: Modeling in the e-Business Era*, edited by D. Simchi-Levi, S. D. Wu, and Z.-J. Shen. Kluwer Academic Publishers.

Chen, Z.-L. (2010). Integrated Production and Outbound Distribution Scheduling: Review and Extensions. *Operations Research, 58*(1), 130–48. https://doi.org/10.1287/opre.1080.0688.

Cheng, B. Y., Leung, J. Y., & Li, K. (2017). Integrated scheduling on a batch machine to minimize production, inventory and distribution costs. *European Journal of Operational Research*, *258*(1), 104-112. https://doi.org/10.1016/j.ejor.2016.09.009.

Darvish, M, Coelho, L.C. (2018). Sequential Versus Integrated Optimization: Production, Location, Inventory Control, and Distribution. *European Journal of Operational Research, 268*(1), 203–14. https://doi.org/10.1016/j.ejor.2018.01.028.

Devapriya, P., Ferrell, W., & Geismar, N. (2017). Integrated production and distribution scheduling with a perishable product. *European Journal of Operational Research*, *259*(3), 906-916. https://doi.org/10.1016/j.ejor.2016.09.019.

Ehm, J., & Freitag, M. (2016). The Benefit of Integrating Production and Transport Scheduling. *Procedia CIRP, 41*, 585–90. https://doi.org/10.1016/j.procir.2015.12.143.

Elmaghraby, S. E. (1978). The Economic Lot Scheduling Problem (ELSP), Review and Extensions. *Management Science, 24*(6), 587–98. https://doi.org/10.1287/mnsc.24.6.587.

Gao, S., Qi, L., & Lei, L. (2015). Integrated batch production and distribution scheduling with limited vehicle capacity. *International Journal of Production Economics*, *160*, 13-25. https://doi.org/10.1016/j.ijpe.2014.08.017.

Jans, R., & Degraeve, Z. (2008). Modeling Industrial Lot Sizing Problems: A Review. *International Journal of Production Research, 46*(6), 1619–43. https://doi.org/10.1080/00207540600902262.

Kergosien, Y., Gendreau, M. & Billaut, J.-C. (2017). A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research, 262* (1), 287–98. https://doi.org/10.1016/j.ejor.2017.03.028.

Lenstra, J. K., & A. H. G. Rinnooy Kan. (1981). Complexity of Vehicle Routing and Scheduling Problems. *Networks, 11* (2), 221–27. https://doi.org/10.1002/net.3230110211.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43-58. https://doi.org/10.1016/j.orp.2016.09.002.

Maxwell, W. L. (1964). The Scheduling of Economic Lot Sizes. *Naval Research Logistics Quarterly, 11*(2), 89–124. https://doi.org/10.1002/nav.3800110202.

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: a review and discussion. *Computers & Industrial Engineering*, *104*, 224-245. https://doi.org/10.1016/j.cie.2016.12.010.

Nawaz, M., Enscore, E. E. & Ham, I. (1983). A Heuristic Algorithm for the m-Machine, n-Job Flow-Shop Sequencing Problem. *Omega, 11*, 91–95.

Ng, C. T., & Lu, L.. (2012). On-Line Integrated Production and Outbound Distribution Scheduling to Minimize the Maximum Delivery Completion Time. *Journal of Scheduling, 15*(3), 391–98.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. 3rd ed. Springer Publishing Company, Incorporated.

Ramezanian, R., Mohammadi, S., & Cheraghalikhani, A. (2017). Toward an integrated modeling approach for production and delivery operations in flow shop system: Trade-off between direct and routing delivery methods. *Journal of Manufacturing Systems, 44*, 79–92. https://doi.org/10.1016/j.jmsy.2017.05.005.

Rohmer, S., & J. C. Billaut. (2015). Production and Outbound Distribution Scheduling: A Two-Agent Approach. In *2015 International Conference on Industrial Engineering and Systems Management (IESM)*, 135–44. https://doi.org/10.1109/IESM.2015.7380148.

Ruiz, R., & Stützle, T. (2007). A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem. *European Journal of Operational Research, 177*(3), 2033–49. https://doi.org/10.1016/j.ejor.2005.12.009.

Russell, R., W. Chiang, and D. Zepeda. (2008). Integrating Multi-Product Production and Distribution in Newspaper Logistics. *Computers & Operations Research, 35*, 1578–88.

Scholz-Reiter, B., Makuschewitz, T., Novaes, A. G., Frazzon, E. M., & Lima Jr, O. F. (2011). An approach for the sustainable integration of production and transportation scheduling. *International Journal of Logistics Systems and Management*, *10*(2), 158-179. https://doi.org/10.1504/IJLSM.2011.042626.

Taillard, E. (1993). Benchmarks for Basic Scheduling Problems. *European Journal of Operational Research, 64*(2), 278–85. https://doi.org/10.1016/0377-2217(93)90182-M.

Tavares Neto, R. F. (2016). Integrating Scheduling and Distribution: Algorithms and Insights. In *Anais Do XLVIII SBPO*, 84–94. Vitória, Espirito Santo, Brazil: Simpósio Brasileiro de Pesquisa Operacional.

Tavares-Neto, Roberto F., & Marcelo Seido Nagano. (2018). An Iterated Greedy Approach to Integrate Production by Multiple Parallel Machines and Distribution by a Single Capacitated Vehicle. *Swarm and Evolutionary Computation*, *44*, 612-621. https://doi.org/10.1016/j.swevo.2018.08.001.

Ulrich, C. A. (2013). Integrated Machine Scheduling and Vehicle Routing with Time Windows. *European Journal of Operational Research, 227*, 152–65.

Viergutz, C., & Knust, S. (2014). Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research*, *213*(1), 293-318. https://doi.org/10.1007/s10479-012-1197-z.