APEM journal

Advances in Production Engineering & Management
Volume 18 | Number 1 | March 2023 | pp 19–31
https://doi.org/10.14743/apem2023.1.454

ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

A matheuristic approach combining genetic algorithm and mixed integer linear programming model for production and distribution planning in the supply chain

Guzman, E.[a,*], **Poler, R.**[a], **Andres, B.**[a]

[a]Research Centre on Production Management and Engineering (CIGIP), Universitat Politècnica de València (UPV), Alcoy, Alicante, Spain

ABSTRACT

A number of research studies has addressed supply chain planning from various perspectives (strategical, tactical, operational) and demonstrated the advantages of integrating both production and distribution planning (PDP). The globalisation of supply chains and the fourth industrial revolution (Industry 4.0) mean that companies must be more agile and resilient to adapt to volatile demand, and to improve their relation with customers and suppliers. Hence the growing interest in coordinating production-distribution processes in supply chains. To deal with the new market's requirements and to adapt business processes to industry's regulations and changing conditions, more efforts should be made towards new methods that optimise PDP processes. This paper proposes a matheuristic approach for solving the PDP problem. Given the complexity of this problem, combining a genetic algorithm and a mixed integer linear programming model is proposed. The matheuristic algorithm was tested using the Coin-OR Branch & Cut open-source solver. The computational outcomes revealed that the presented matheuristic algorithm may be used to solve real sized problems.

ARTICLE INFO

Keywords:
Production and distribution planning;
Supply chain;
Matheuristic;
Genetic algorithm;
Mixed integer linear programming model

**Corresponding author:*
eguzman@cigip.upv.es
(Guzman, E.)

Article history:
Received 29 April 2022
Revised 21 December 2022
Accepted 27 December 2022

1. Introduction

The globalisation of markets has lee to companies to optimise their processes and resources to remain competitive. Nowadays, optimisation is a relevant factor for improving firms' performance, and for turning the challenges that they face into competitive advantages [1]. One optimal strategy for profits that can minimise a company's total costs is to integrate different business functions, such as purchasing, inventory management, production and distribution [2]. Therefore, it is an important factor for optimizing supply chain enterprises to establish greater integration production and distribution planning (PDP) [3]. The PDP problem is defined by Safaei *et al.* [4] as a firm's scheduling process to manufacture the right products and to ship the quantities of the right products to the right place at the right time.

Increasing pressure to minimise total production and logistic costs means that supply chain agents are having to re-examine production-distribution policies, and to maximise the use of

physico-technological assets [5]. Properly coordinating PDP in supply chains is a challenging problem because companies expand internationally and move to a competitive environment that requires greater collaboration. Efficient supply chain cooperation involves many coordinated decisions being made at several decision levels (e.g., strategical, tactical, operational) about products, financial resources and information.

In this context, the present research work develops an efficient matheuristic approach to solve the integrated PDP problem. A matheuristic algorithm is defined by Boschetti *et al*. [6] as "the interoperation of metaheuristics and mathematical programming techniques". There are different approaches for combining metaheuristics with exact methods, and each technique has its individual advantages and disadvantages. However, the aim is to benefit from synergy. Several researchers present a taxonomy that classifies this type of cooperation. Table 1 shows several approaches by different authors, although some have similar characteristics.

When designing a matheuristic, the question is, which components can work together to generate an efficient algorithm? Although supplying a collaboration rule does not seem a feasible approach, the matheuristic design involves functionality and architecture. Thus, the cooperation level can be ranked according to its hierarchy, as in Table 2.

Table 1 Classification of matheuristic approaches

Approach	Classes	
Cooperation between exact and local search methods [7] based on Dumitrescu & Stützle [8]	• Exact algorithms to browsing through neighbourhoods in local search algorithms. • Exact algorithms intended for specific hybrid metaheuristics procedures. • Explore boundaries in constructive heuristics. • Local searches or constructive algorithms guided by data from integer programming model relaxations. • Exact algorithms for smaller problems using solutions from local searches	
Combination between exact techniques and metaheuristic algorithms [9], [10]	Collaborative combinations Algorithms exchange information. None is contained in any other. Both procedures can be executed sequentially, interlaced or in parallel.	Subclasses • Sequential execution • Parallel and intertwined execution
	Integrative combinations One technique is component-integrated into another technique with a master-slave structure. An exact or metaheuristic algorithm can be presented with a master-type structure and at least one integrated slave.	• Incorporating exact algorithms into metaheuristics • Incorporating metaheuristics into exact algorithms
"MASTER-SLAVE" structure with a guiding process and application process [11]	• Metaheuristics operate at the master level and, thereby, control and guide actions to the exact technique. • The exact method operates as a master to call/control by the metaheuristic approach.	

Table 2 Hierarchical matheuristic classification [12]

Hierarchy	Description
LRH (low-level relay hybrid)	It depicts hybrid schemes in which a metaheuristic approach is included in an exact approach to improve the search strategy
LTH (low-level teamwork hybrid)	It describes one search element of a metaheuristic to be replaced with another exact algorithm
HRH (high-level relay hybrid)	Autonomous algorithms are executed in a sequence. The stage can be either pre-processing or post-processing, i.e., two groups of algorithms (metaheuristics + exact algorithms) are provided with some data in sequence
HTH (High-level Teamwork Hybrid)	A combination of metaheuristics and exact algorithms that performs a parallel search and cooperate to find relaxed optimal solutions, better lower or upper bounds, optimum subproblem solutions, partial solutions, etc. Metaheuristics and exact algorithms solve partial, specialised or global optimisation problems and exchange helpful information.

Recent studies, such as that presented by Kumar *et al.* [13] , provide a literature review of the quantitative approaches applied to combined PDP. These authors concluded that the main modelling approaches for this problem type are MILP (mixed integer linear programming), while the

main applied solution approaches are those that resort to optimisation software, followed by genetic algorithms (GAs). Computational experiments in small instances use mainly LINGO and CPLEX to solve MILP and Matlab and C++ in large instances to solve heuristic and metaheuristic methods. In this review, we observe that matheuristic methods have not yet been discussed in-depth. As far as the authors know, to date no research has addressed the PDP problem using this type of matheuristic.

In this context, we propose developing a solution strategy for the PDP problem with a mathematical algorithm that is positioned in the hierarchical classification described in Table 2 as a High-level Teamwork Hybrid. This strategy is useful because the search space of the MILP model is considered to be too big for a solver to solve it. Therefore, we employ a GA that exchanges information in parallel to the MILP model to diminish the search space.

Given the complexity of PDP problems, they prove difficult when implementing large datasets or solving real SME problems with a MILP model. For this reason, some companies choose to use commercial solvers for this type of problem. However, some SMEs cannot afford to buy a commercial solver because of its high cost but, as digitisation needs are accelerating, many companies are considering how to be equipped with a digital infrastructure insofar as it does not constrain them and does not cost too much. So those SMEs that have implemented open-source software have made significant savings in technology spending because they do not have to pay annual software licences and have not run the risk of software becoming obsolete when licences expire [14].

Accordingly, this article contributes: (i) a new matheuristic approach to solve the PDP problem; (ii) the matheuristic algorithm was tested and compared to a non-commercial Coin-Branch & Cut (CBC) solver and employs a free open-source operating system (Linux). The proposed approach's effectiveness is proven by solving randomly generated test datasets with real data sizes.

The rest of this article is arranged as follows. Section 2 briefly presents a literature review about the integrated approach to supply chain PDP. Section 3 offers a mathematical model. Section 4 details the matheuristic algorithm for solving the planning-distribution problem. Section 5 presents the evaluation of the matheuristic algorithm using large instances to simulate real-life companies. Finally, Section 6 defines some conclusions and future research directions.

2. Related works

This section reviews the literature about integrating decisions from PDP functions, along with the solution approaches suggested for these problems. This problem has been paid plenty of attention in recent years. Literature reviews like that by Chen [15] indicate several future research lines. One of them states that more effort should be made to create heuristic or metaheuristic methods for this type of problems, which are NP-hard, as there are very few solution algorithms for this type of problems. Years later Fahimnia *et al*. [16] describe that the use of heuristic, metaheuristic and simulation techniques predominate in the literature, but propose employing new techniques, and suggest having to extend the effectiveness of solution techniques to deal with realistic PDP problems as most techniques have been applied to deal with small- and medium-sized problems. Lastly, the work by Kumar *et al*. [13] indicates the extensive use of metaheuristic algorithms like heuristic algorithms, GAs and exact methods, but does not reveal the use of matheuristic algorithms.

Accordingly, related work like that of Raa *et al.* [5] proposes an aggregate PDP model for injection moulding production in the many facilities of a plastics manufacturer. This MILP is solved by the Gurobi solver for small instances. For large instances, these authors employ an iterative matheuristic that utilises a decomposition heuristic. Bilgen and Çelebi [1] offer a combined simulation and MILP approach for integrated production and distribution problems in the dairy industry. The MILP model is solved with CPLEX and the hybrid approach employs ARENA.

Su *et al.* [17] propose combining distinct algorithms like the GA and particle swarm optimisation (PSO). The GA comes with a learning scheme, and a hybrid algorithm that combines PSO techniques with the GA and a learning scheme to solve both partner selection and the PDP problem in a manufacturing chain design. Moattar Husseini *et al.* [18] put forward bi-objective MILP for integrated PDP with manufacturing partners. One of the objectives of this model is to minimise the

total cost by covering production, inventory holding purchases from partners and transport-distribution costs. Another objective aims to maximise the quality level of the products that partners supply on the planning horizon. For this problem, they employ LINGO to solve the model in small instances. However, as the problem in large instances is classified as NP-hard, the authors solve it by a Non-Dominant GA II (NSGA-II) and a Multi-Objective PSO (MOPSO) algorithm. The computational results confirm the suitability and practicality of these two algorithms, but the MOPSO algorithm obtains better results in most instances. Devapriya *et al.* [19] report a PDP problem with a perishable product. The problem is modelled by MILP, is solved with CPLEX, employs a memetic algorithm to solve the problem in large instances, and obtains good solutions in a relatively shorter computational time.

Kazemi *et al.* [3] put forward a hybrid algorithm that combines a multi-agent system and three metaheuristic algorithms, including a GA, a tabu search and simulated annealing. They propose a MILP model that is solved with LINGO. They employ Matlab to evaluate the hybrid approach. Their results reveal that LINGO better works in small instances, while the hybrid approach delivers better solutions in large instances. In a multifactory supply chain, Gharaei and Jolai [20] study a multi-agent scheduling problem with distribution decisions. To do so, they propose using a MILP formulation to solve the problem with CPLEX by employing small and medium instances. They also develop a multi-objective evolutionary algorithm based on decomposition by combining the Bees algorithm and using Matlab, which well performs in long instances. Marandi and Fatemi Ghomi [21] put forward an integrated production-distribution scheduling problem. They aim to simultaneously find a production schedule and a vehicle routing solution to minimise the sum of delay and transportation costs. They apply CPLEX for small problems and propose a new algorithm for medium and large problems, namely the Improved Imperialist Competitive Algorithm, which applies a local search algorithm based on a simulated annealing algorithm.

The literature review highlights a growing research tendency to integrate PDP functions. It reveals that companies tend to collaborate with manufacturing partners to better respond to demanding market conditions, and they focus more on their core activities. Interest is shown in heuristic and metaheuristic methods, which are frequently employed to solve these problems with large instances. These instances normally represent the size of the data actually employed by real companies, although different variants of the PDP problem exist. Models tend to be solved mostly with a commercial solver, of which CPLEX is the most widespread. Despite previous works having discussed some combinations of the above algorithms, other combinations have not yet been addressed by the literature, such as those using matheuristic algorithms in practice.

Based on these results, this study considers an integrated PDP problem formulated as a MILP model. As the literature reports the potential effectiveness of GA-based algorithms [18], a combined solution approach with a GA and a mathematical model is herein considered. A non-commercial solver and an open-source operating system are also implemented. The next sections discuss the particulars of the posed problem, its formulation and the solution approach.

3. Problem definition

This section offers details of the studied problem and formulates the proposed model. The PDP problem herein contemplated is based on Park [22].

The MILP model takes these assumptions:

- For the production stage: many production plants produce multiple items with a limited capacity per time period. Each product type has a setup cost, while production plants have a limited storage capacity, and produced items are shipped directly to points of sale.
- For the distribution stage: distribution is performed with a fleet of homogeneous vehicles, which are parked in production plants.
- The vehicle movement incurs on: (i) a fixed cost in relation to the depreciation of vehicles, insurance, etc.; (ii) a variable cost according to the transported item, quantity and route.

For points of sale: an item's demand during a period at a point of sale consists of two components: (i) "core demand": the amount of main demand that the point of sale must meet by loyal customers in the long term; (ii) "forecasted demand": the total amount, including core demand. Unmet demand at a point of sale is considered a stockout (rejected demand) and does not allow deferred demand. Each point of sale can maintain a limited amount of inventory at a very high cost. An overview of the considered problem is demonstrated in Fig. 1.

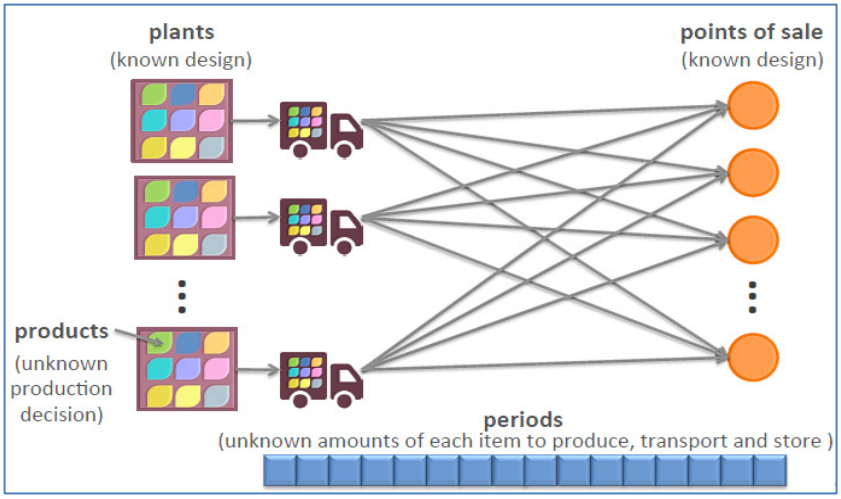

Fig 1 Illustration of the integrated production and distribution planning problem

3.1 Notation

The PDP problem nomenclature is shown below.

Table 3 The MILP model nomenclature

Notation	Description
Sets	
i	Index of plants $i \in \{1,..., I\}$
j	Index of points of sale, $j \in \{1,..., J\}$
k	Index of products (parts) $k \in \{1,..., K\}$
t	Index of time periods $t \in \{1,..., T\}$
Parameters	
C_{ik}	cost of producing 1 unit of product k in plant i
s_{ik}	cost of setup for product k in plant i
o_{ik}	time of producing 1 unit of product k in plant i
u_{ik}	time of setup for product k in plant i
hp_{ik}	unit holding cost per period for product k in plant i
L_i	production capacity per period of plant i
$dijk$	cost of transporting 1 unit of product k from plant i to point of sale j
g	fixed cost per vehicle
B	capacity per vehicle
E_{jkt}	core demand of product k at point of sale j during period t
F_{jkt}	forecasted demand of product k at point of sale j during period t
p_{jk}	price of sale of 1 unit of product k at point of sale j
hr_{jk}	unit holding cost per period for product k at point of sale j
Wr_j	inventory capacity of point of sale j
v_{jk}	unit stockout cost of item k at point of sale j
Variables	
x_{ikt}	amount of product k produced at plant i during period t
q_{ijkt}	amount of product k transported from plant i to point of sale j during period t
Y_{ikt}	1 if product k is produced a plant i during period t; 0 otherwise
ap_{ikt}	inventory level of product k at plant i during period t
ar_{jkt}	inventory level of product k at point of sale j during period t
z_{ijt}	number of vehicles needed for distribution from plant i to point of sale j during period t

The objective function maximises sales revenues at points of sale, minus the costs of setup, production and inventory at plants, the costs of inventory and stockout at points of sale, and the costs of vehicles and transport.

$$
\begin{aligned}
MaxZ = & \sum_j \sum_k p_{jk} \cdot \sum_t \left(ar_{jkt-1} + \sum_i q_{ijkt} - ar_{jkt} \right) \\
& - \left(\sum_i \sum_k c_{ik} \cdot \sum_t x_{ikj} + \sum_i \sum_k S_{ik} \cdot \sum_t Y_{ikt} + \sum_i \sum_k hp_{ik} \cdot \sum_t ap_{ikt} + \right) \\
& - \left(\sum_j \sum_k hr_{jk} \cdot \sum_t ar_{jkt} + \sum_j \sum_k v_{jk} \cdot \sum_t \left(F_{jkt} - ar_{jkt-1} \right. \right. \\
& \left. \left. + \sum_t q_{ijkt} + ar_{ijkt} \right) \right) - \left(g \cdot \sum_i \sum_j \sum_t z_{ijt} + \sum_i \sum_j \sum_k d_{ijk} \cdot \sum_i q_{ijkt} \right)
\end{aligned} \tag{1}
$$

Subject to:

Material flow constraints

$$ap_{ikt} = ap_{ikt-1} + x_{ikt} - \sum_j q_{ikt} \qquad \forall i,k,t \tag{2}$$

$$ar_{jkt-1} + \sum_i q_{ijkt} - ar_{jkt} \geq E_{jkt} \qquad \forall j,k,t \tag{3}$$

$$ar_{jkt-1} + \sum_i q_{ijkt} - ar_{jkt} \leq F_{jkt} \qquad \forall j,k,t \tag{4}$$

Constraint Eq. 2 guarantees the inventory of all products in each plant at the end of every period. Constraint Eq. 3 ensures meeting "core demand" at each point of sale per product during each time period. Constraint Eq. 4 ensures that the demand served for any product at any point in time at any point of sale never exceeds the expected demand ("forecast demand").

Physical resource limitations

$$\sum_k o_{ik} \cdot x_{ikt} + \sum_k u_{ik} \cdot Y_{ikt} \leq L_i \qquad \forall i,t \tag{5}$$

$$x_{ikt} <= M \cdot Y_{ikt} \qquad \forall i,k,t \tag{6}$$

$$\sum_k ar_{jkt} \cdot \leq Wr_j \qquad \forall j,t \tag{7}$$

$$z_{ijt} \sum_k \frac{q_{ijkt}}{B} \qquad \forall i,j,t \tag{8}$$

$$ap_{ik0} = 0, ar_{ik0} = 0 \qquad \forall i,j,k \tag{9}$$

Constraint Eq. 5 guarantees that, per plant during each period, the capacity consumption due to the processing and preparation times of processed items never exceeds the plant's available production capacity. Constraint Eq. 6 ensures that if a quantity of a certain product is produced in a plant during a period, a setup of this product is necessary. Constraint Eq. 7 ensures that the amount of products stored at a point of sale during every period must never exceed the point of sale's storage capacity. Constraint Eq. 8 computes the number of vehicles required to transport products from every plant to each point of sale during all periods. Constraint Eq. 9 represents the initial inventory levels in plants and at points of sale.

$$Y_{ikt} \in \{0,1\} \qquad \forall i,k,t \tag{10}$$

$$x_{ikt}, q_{ijkt}, ap_{ikt}, ar_{ikt}, z_{ijt} \in \mathbb{Z} \qquad \forall i,j,k,t \tag{11}$$

Constraints Eq. 10 and Eq. 11 indicate the binary nature of Y_{ikt} and the integer nature of some variables.

4. Proposed matheuristic solution method

The PDP problem is a complex one to solve given the number of integer variables that corresponds to produced and transported products, the inventory level in the plant and at points of sale, and the vehicles needed for distribution, plus the binary variable that indicates in which plants products are produced. Given the difficulty of this problem, a solution methodology is offered and describes how the GA is combined with the MILP model to evaluate the solutions for the PDP problem.

4.1 Initial population

Each individual in the population corresponds to binary decision variable Y_{ikt}, which takes 1 if product *k* is produced in plant *i* during period *t*, and 0 otherwise. An individual's length is a one-dimensional matrix of size $I \times K \times T$ (multiplication of the quantities of every index). The population takes a binary structure and is generated randomly from a uniform distribution with a 50 % probability of 1 appearing on an individual's chromosome. The computational results indicate that fewer infeasible individuals are generated if this probability is applied. Population size N_{pop} equals 10, so we employ this small size to increase the GA's speed. Koljonen and Alander [23] confirm that a small population size increases the optimisation speed to a certain extent. We prove that using this population size suffices to obtain good solutions.

4.2 Evaluation function

The fitness function measures the quality of an individual in the population. The problem looks for solutions that maximise the benefits that the objective function represents. The PDP problem's computational difficulty focuses mostly on binary variable Y_{ikt}, which refers to the decisions made about which product to produce in which plant. This means that the GA is in charge of producing a suitable binary chromosome with equal dimensions to the binary variable.

As this binary chromosome corresponds to each individual in the population, the evaluation of each individual is made by formulating the mathematical model. The computational and execution times of a MILP versus a linear programming (LP) model are longer given the SIMPLEX algorithm's computational efficiency versus the algorithms dedicated to solve problems with integer or binary variables, along with the problem's difficulty, which is considered NP-hard. The proposed MILP model comprises one binary variable and five integer variables. Thus, to improve matheuristic performance, we apply MILP model relaxation. The MILP relaxation to obtain LP is given by transforming integer variables into continuous variables, and by transforming the binary variable into data.

At this time the solver is in charge of solving LP and the GA is responsible for supplying the binary variable. The binary variable of the GA is fixed to LP. Thus, when executing the matheuristic, it can be quickly solved even for very large problems. In our experiments, on average LP obtains better results than MILP by 3.84 %. Thus, to obtain a final result, we employ the best binary chromosome obtained during the evaluation process and launch MILP to gain a final result. This is explained in Section 4.6.

4.3 Selection

In the selection stage, a set of individuals from the current population is chosen to be used as the parents for the crossing stage. The roulette wheel approach [24] is taken to select the individuals with the best fitness values in accordance with the uniform probability of selection distributed over the range [0...1], and the worst individuals are eliminated from one generation to another so that the best individuals are more probably selected.

4.4 Crossover

The single point crossover technique [25] is applied. Two parents (P1 and P2) are selected by the fortune roulette wheel selection technique. Then the P1 and P2 chromosomes are cut at a point that is randomly generated and a new offspring (OF) is generated with the genetic information of its parents, as illustrated in Fig. 2.

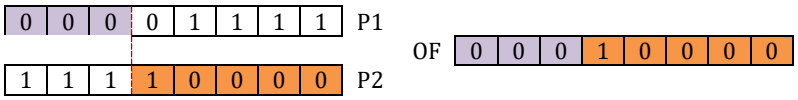

Fig 2 Single-Point Crossover

4.5 Mutation

Mutation allows the GA to explore a bigger region of the ranges of potential solutions by including random genetic changes, which are produced by introducing variations into individuals, and thus allowing the GA to not fall into local optima [26]. The swap mutation operator is implemented here. This mutation method randomly selects two genes from offspring and then exchanges the gene content in its offspring, as shown in Fig. 3.

The offspring that undergo mutation are selected with uniform probability *Pm*=1. This means that all offspring are mutated but, in order not to lose the normal offspring, both the normal and mutated offspring remain in the resulting population. This avoids losing a good solution obtained by the crossing process as a mutation can provide a worse fitness value [27]. Then the two offspring (normal and mutated) are included in the population by replacing the two worst individuals in that population to leave a constant population size.

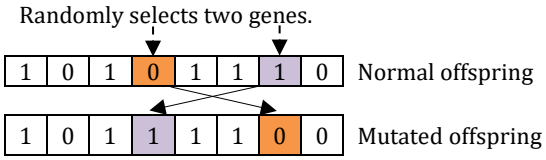

Fig. 3 The Swap mutation operator

4.6 The matheuristic approach procedure

The best individual with the best fitness is selected at the end of the calculation time (stopping criterion) of the matheuristic GA. With this binary chromosome, MILP is launched.

In the evaluation function, MILP is relaxed to LP to reduce computational effort and to obtain a sufficiently good solution. To obtain a definitive solution, MILP is used, i.e., by removing relaxation and employing the binary chromosome of the individual with the best fitness provided by the GA.

The binary chromosome is set at the Y_{ikt} variable of MILP. This means that the binary chromosome becomes a parameter for the model. Then the MILP model is launched, the solver solves the integer variables and the GA provides the binary variable

The advantage of using matheuristic, and not directly using the solver, lies in the search space of the MILP model being significantly reduced when a matheuristic is employed to deal with binary variables.

5. Results and discussion: Numerical experiment case study

In the present section, a set of synthetic data is used to evaluate our approach. In this type of problem, real data sets are generally large, which renders it unsolvable with many plants, products, outlets and periods. To assess how the matheuristic and the non-commercial solver perform, in computational tests we apply large instances, which are randomly generated according to the outlined parameters and formulations in Park [22]. To create these data, we created a synthetic data generator, which appears at: https://bit.ly/synthetic_data_generator.

Park [22] analysed large datasets with similar characteristics to those in Table 4. Park used CPLEX to solve MILP but did not present any results for large instances because of the problems' computational difficulty, which is why he applied this solver only for small instances. We use the same data for plant size (5), points of sales (from 40 to 65), products (5) and periods (from 10 to 12), as presented in the above-mentioned study.

The software followed in this research is a non-commercial optimisation solver from the Computational Infrastructure for Operation Research (COIN-OR) community called COIN-OR Branch

and Cut Solver (CBC) [28]. This open-source solver is generally employed for MILP problems. The MILP model and matheuristic were implemented in Python with the Pyomo package [29]. Experiments were run by an Intel Core i7 2.80 GHz processor (6 GB RAM) in an Ubuntu 20.04.1 LTS operating system.

The performances of the matheuristic approach and the proposed MILP through computational experiments were compared to one another to identify the best performing method. The resulting GAP of the MILP solved by CBC and the matheuristic is calculated as indicated in Equation (12).

$$GAP(\%) = \frac{|UB - Best_{sol}|}{|Best_{sol}|} \tag{12}$$

Where UB indicates the upper solver bound, and $Best_{sol}$ refers to the best solution generated by either the mathematical model or the matheuristic approach.

5.1 Experimental results

In order to demonstrate the proposed approach's efficiency and performance, the computational experiments with different large instances are provided. Table 4 compares the solution's efficiency among the solutions obtained by solving MILP with CBC and the matheuristic one with CBC. The first column in this table denotes the name of the instance, followed by the number of plants (I), points of sale (J), products (K) and periods (T). For all the instances, the applied criterion is the same calculation time that corresponds to 14,400 seconds. We executed the matheuristic algorithm 20 times for each instance in order to evaluate and avoid atypical performance.

The MILP solved by CBC obtained solutions for two (I2, I4) of the six instances, but it was unable to find optimal or good solutions. The matheuristic gave good solutions for all the instances. Fig. 4 illustrates the total profit obtained by matheuristic and CBC. I2a and I4a show how the matheuristic approach evolves and converges towards good solutions, along with how CBC performs at around 7,200 computation seconds, while I4b and I4b show the behaviour of both the matheuristic and CBC at 14,400 processing seconds. For the I2 instance, CBC gave a feasible solution at 5,152.76 seconds (see Fig. 4) with GAP = 17.10 %. GAP improved up to 14,400 seconds by 0.02 %. For the matheuristic for the same instance, it obtained feasible solutions from 71.05 seconds, with GAP less than 10 % at 1,880 seconds (see Fig. 5).

Table 4 Performance comparison between CBC and Matheuristic

Instance	Problem				CBC			Matheuristic	
	I	J	K	T	Total profit	Upper bound	GAP	Total profit	GAP
I1	5	40	5	12	Unfeasible solution found	5746740.5	-	5117847	12.28 %
I2	5	50	5	10	5873321	6876383.6	17.08 %	6347862	8.33 %
I3	5	45	5	12	Unfeasible solution found	6785206.6	-	6157783	10.18 %
I4	5	60	5	10	6643133	8037979.9	21.00 %	7487111	7.36 %
I5	5	50	5	12	Unfeasible solution found	6932259.4	-	6294622	10.13 %
I6	5	65	5	10	Unfeasible solution found	8272162.9	-	7640947	8.26 %

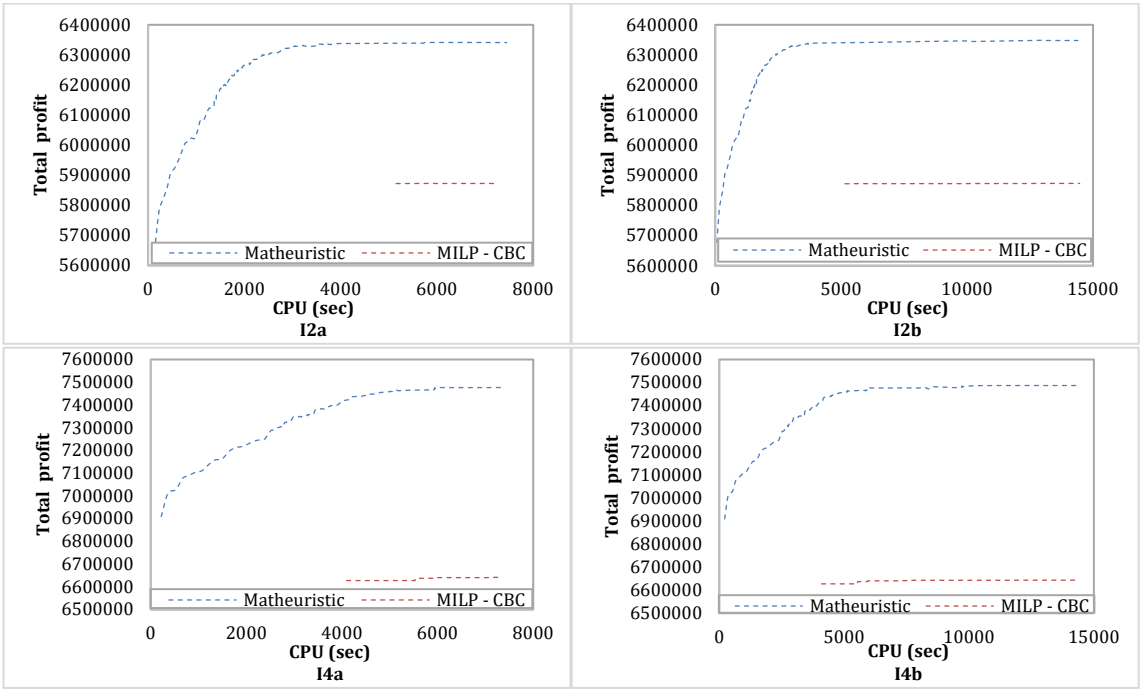

Fig 4. Time spent by matheuristic to find a feasible solution

With the I4 instance, CBC performance visibly improves. It obtains solutions in 4,097 seconds and becomes the best solution in 5,538 seconds (see Fig. 5). Matheuristic better performs than CBC by reaching feasible solutions in shorter computational times and reaches a GAP below 10 % after 2,735 seconds (see Fig. 5). This means that matheuristic outperforms CBC by 13.64 %.

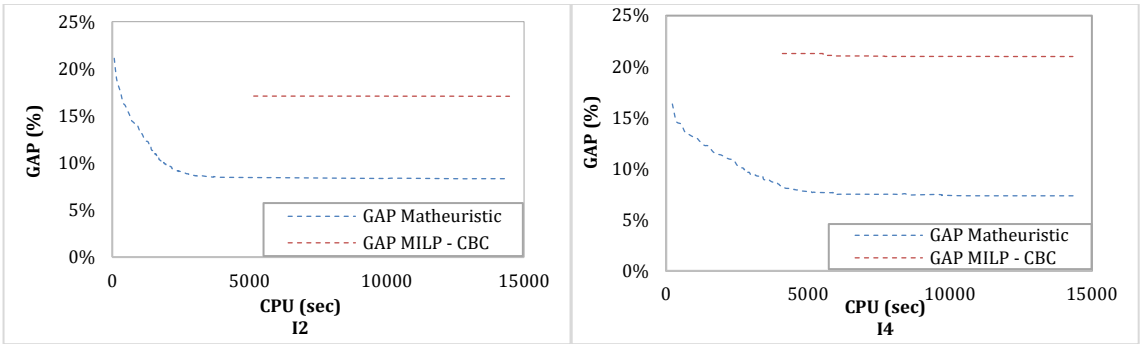

Fig. 5 Comparison of matheuristic and MILP-CBC performance

The complexity of the instances and the size of the problem mean that CBC is unable to find feasible solutions. Nevertheless, the combination of a GA with CBC gives better results with feasible solutions in shorter computational times. A matheuristic's efficiency is linked with the solver's speed because the solver is in charge of evaluating solutions by the GA's evaluation function. Moreover, as the evaluation function is the principal component of GAs [30], employing a non-commercial solver combined with a GA offers good results, as herein shown, and the matheuristic is more efficient in solving problems with many variables and parameters, and can be a useful alternative for large instances. When utilising a non-commercial solver like CBC, a matheuristic can support the solver to find better solutions.

In order to further demonstrate the efficiency of the proposed matheuristic, we compare it to Gurobi 9.1.1, i.e., the MILP and LP of the matheuristic are solved with Gurobi. We employ the same aforementioned computational conditions and apply a processing time of 14,400 seconds. The computational results given by Gurobi are better than those of CBC. Thus Gurobi obtains feasible solutions in all the instances in much shorter solution times. However, the matheuristic is better for achieving a lower GAP than Gurobi in all instances (see Fig. 6).

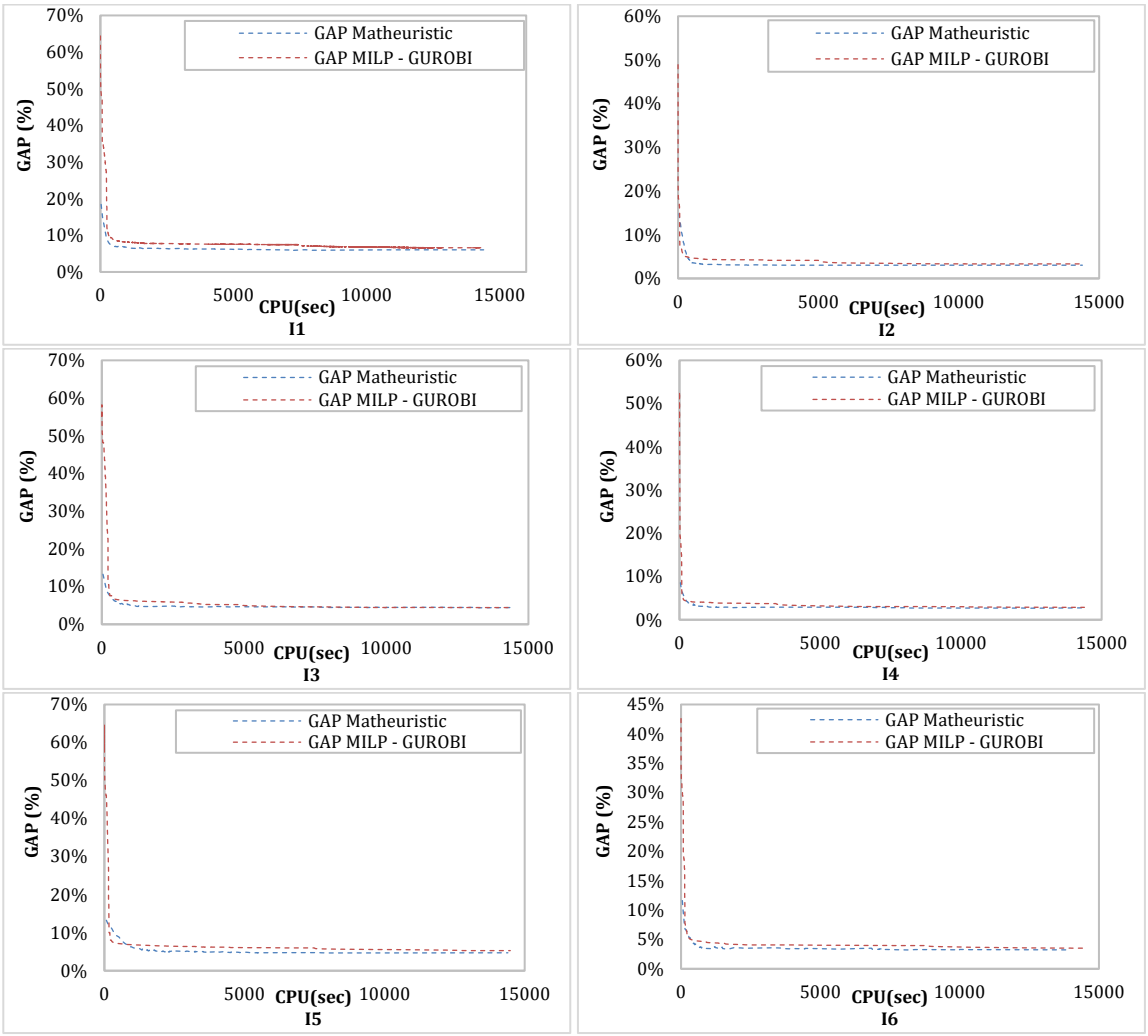

Fig. 6 Comparison of matheuristic and Gurobi performance

6. Conclusion

The PDP problem has long since been studied for the practical applications that it can offer industry. One such case is enterprises with different manufacturing plants in several locations, perhaps in the same city or country, or in others, which must decide the amount of products to be produced in plants, the quantity of products to be stored in plants during each period, the number of products to deliver to various points of sale according to product demand and the inventory of finished products at points of sales. Although several resolution techniques have been used for this problem and its variations, heuristic and metaheuristic algorithms can provide excellent results as combined or hybrid approaches. Likewise, combinations between metaheuristic techniques and exact approaches can offer better results for real-life problems because these combinations make the most of the benefits of both techniques [7]. In this context, the present paper intends to solve the PDP problem in real-world large data sizes. The problem is modelled as MILP, and a matheuristic solution approach is presented that combines a GA and an LP model.

Computational tests were performed on a large dataset capable of simulating real-world problems. The development of this approach stems from SMEs having to use open-source tools and the need to digitise companies because they must compete in today's market. Many SMEs cannot have access to software with paid licenses, due to the high-costs they may have to adapt the software to the needs of the enterprises. The main research contribution is about applying a matheuristic approach by employing a non-commercial solver (CBC). We also tested the performance of the non-commercial solver with an NP-hard MILP model. The computational tests run on different

instances showed that our approach offers markedly improved results than the exact method. Matheuristic obtained competitive results in a short time. When solving MILP, CBC is unable to acquire feasible solutions for four of the six computed instances. However with our proposed matheuristic, and by also using CBC for solving relaxed LP, our results were good for all instances. Matheuristic can perform better even when using a commercial solver like Gurobi. Therefore, matheuristic can offer a real technical and economical application and is affordable mainly for SMEs that cannot pay a commercial solver or do not recurrently resort to one. This approach is feasible thanks to the proposed model's simplicity. The matheuristic also offers the benefit of making the most of the solver's features, regardless of them being commercial or not, because the matheuristic improves the solver's performance. The proposed approach has the limitation that its effectiveness depends on the selected solver, since the solutions of the matheuristic with a commercial solver (Gurobi) are better than those obtained with a non-commercial solver (CBC).

Other metaheuristics can be used for future work, such as memetic algorithms, ant colony optimisation or tabu search, and other highly complex problems can also be tested. Other genetic operators can be evaluated, or specific heuristics can be used to improve the GA's performance.

Acknowledgement

This work was supported by the Conselleria de Educación, Investigación, Cultura y Deporte - Generalitat Valenciana for hiring predoctoral research staff with Grant (ACIF/2018/170) and European Social Fund with Grant Operational Program of FSE 2014-2020, the Valencian Community. The research leading to these results received funding from the European Union H2020 Programme with grant agreement No. 958205 "Industrial Data Services for Quality Control in Smart Manufacturing" (i4Q) and the Regional Department of Innovation, Universities, Science and Digital Society of the Generalitat Valenciana entitled "Industrial Production and Logistics Optimization in Industry 4.0" (i4OPT) (Ref. PROMETEO/ 2021/065)

References

[1] Bilgen, B., Çelebi, Y. (2013). Integrated production scheduling and distribution planning in dairy supply chain by hybrid modelling, *Annals of Operations Research,* Vol. 211, No. 1, 55-82, doi: 10.1007/s10479-013-1415-3.

[2] Armentano, V.A., Shiguemoto, A.L., Løkketangen, A. (2011). Tabu search with path relinking for an integrated production-distribution problem, *Computers & Operations Research*, Vol. 38, No. 8, 1199-1209, doi: 10.1016/j.cor.2010.10.026.

[3] Kazemi, A., Zarandi, M.H.F., Azizmohammadi, M. (2017). A hybrid search approach in production-distribution planning problem in supply chain using multi-agent systems, *International Journal of Operational Research*, Vol. 28, No. 4, 506-527, doi: 10.1504/IJOR.2017.082611.

[4] Safaei, A.S., Moattar Husseini, S.M., Farahani, R.Z., Jolai, F., Ghodsypour, S.H. (2010). Integrated multi-site production-distribution planning in supply chain by hybrid modelling, *International Journal of Production Research*, Vol. 48, No. 14, 4043-4069, doi: 10.1080/00207540902791777.

[5] Raa, B., Dullaert, W., Aghezzaf, E.H. (2013). A matheuristic for aggregate production-distribution planning with mould sharing, *International Journal of Production Economics*, Vol. 145, No. 1, 29-37, doi: 10.1016/j.ijpe.2013.01.006.

[6] Boschetti, M.A., Maniezzo, V., Roffilli, M., Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control, In: Blesa, M.J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.), *Hybrid metaheuristics, HM 2009. Lecture notes in computer science, Vol. 5818,* Springer, Berlin, Germany, 171-177, doi: 10.1007/978-3-642-04918-7_13.

[7] Jourdan, L., Basseur, M., Talbi, E.G. (2009). Hybridizing exact methods and metaheuristics: A taxonomy, *European Journal of Operational Research*, Vol. 199, No. 3, 620-629, doi: 10.1016/j.ejor.2007.07.035.

[8] Dumitrescu, I., Stützle, T. (2003). Combinations of local search and exact algorithms, In: Cagnoni, S., Johnson, C.G., Romero Cardalda, J.J., Marchiori, E., Corne, D.W., Meyer, J.-A., Gottlieb, J., Middendorf, M., Guillot, A., Raidl, G.R., Hart, E. (eds.), *Applications of evolutionary computing, EvoWorkshops 2003, Lecture notes in computer science*, Springer, Berlin, Germany, 211-223, doi: 10.1007/3-540-36605-9_20.

[9] Raidl, G. R., Puchinger, J. (2008). Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization, In: Blum, C., Aguilera, M.J.B., Roli, A., Sampels, M. (eds.), *Hybrid metaheuristics. studies in computational intelligence*, Springer, Berlin, Germany, 31-62, doi: 10.1007/978-3-540-78295-7_2.

[10] Puchinger, J., Raidl, G.R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification, In: Mira, J., Álvarez, J.R. (eds.), *Artificial intelligence and knowledge engineering applications: A bioinspired approach, IWINAC 2005, Lecture notes in computer science*, Springer, Berlin, Germany, 41-53, doi: 10.1007/11499305_5.

[11] Caserta, M., Voß, S. (2010). Metaheuristics: Intelligent problem solving, In: Maniezzo, V., Stützle, T., Voß, S. (eds.), Matheuristics. Annals of information systems, Vol 10, Springer, Boston, USA, 1-38, doi: 10.1007/978-1-4419-1306-7_1.
[12] Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning, *Annals of Operations Research*, Vol. 240, No. 1, 171-215, doi: 10.1007/s10479-015-2034-y.
[13] Kumar, R., Ganapathy, L., Gokhale, R., Tiwari, M.K. (2020). Quantitative approaches for the integration of production and distribution planning in the supply chain: A systematic literature review, *International Journal of Production Research*, Vol. 58, No. 11, 3527-3553, doi: 10.1080/00207543.2020.1762019.
[14] Reshad, A., Sinha, S. (2020). Open source software solution for small and medium enterprises, *international journal of computer sciences and engineering*, Vol. 8, No. 6, 86-90.
[15] Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: Review and extensions, *Operations Research*, Vol. 58, No. 1, 130-148, doi: 10.1287/opre.1080.0688.
[16] Fahimnia, B., Farahani, R.Z., Marian, R., Luong, L. (2013). A review and critique on integrated production-distribution planning models and techniques, *Journal of Manufacturing Systems*, Vol. 32, No. 1, 1-19, doi: 10.1016/j.jmsy.2012.07.005.
[17] Su, W., Huang, S.X., Fan, Y.S., Mak, K.L. (2015). Integrated partner selection and production-distribution planning for manufacturing chains, *Computers & Industrial Engineering*, Vol. 84, 32-42, doi: 10.1016/j.cie.2015.01.015.
[18] Moattar Husseini, Z., Karimi, B., Moattar Husseini, S.M., Ghodsipour, S.H. (2015). Multi-objective integrated production distribution planning concerning manufacturing partners, *International Journal of Computer Integrated Manufacturing*, Vol. 28, No. 12, 1313-1330, doi: 10.1080/0951192X.2014.972460.
[19] Devapriya, P., Ferrell, W., Geismar, N. (2017). Integrated production and distribution scheduling with a perishable product, *European Journal of Operational Research*, Vol. 259, No. 3, 906-916, doi: 10.1016/j.ejor.2016.09.019.
[20] Gharaei, A., Jolai, F. (2018). A multi-agent approach to the integrated production scheduling and distribution problem in multi-factory supply chain, *Applied Soft Computing,* Vol. 65, 577-589, doi: 10.1016/j.asoc.2018.02.002.
[21] Marandi, F., Fatemi Ghomi, S.M.T. (2019). Integrated multi-factory production and distribution scheduling applying vehicle routing approach, *International Journal of Production Research*, Vol. 57, No. 3, 722-748, doi: 10.1080/00207543.2018.1481301.
[22] Park, Y.B. (2005). An integrated approach for production and distribution planning in supply chain management, *International Journal of Production Research*, Vol. 43, No. 6, 1205-1224, doi: 10.1080/00207540412331327718.
[23] Koljonen, J., Alander, J.T. (2006). Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms, In: *Proceedings of 9th Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, Espoo, Finland, 54-60.
[24] Zhong, J., Hu, X., Zhang, J., Gu, M. (2005). Comparison of performance between different selection strategies on simple genetic algorithms, In: *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06),* Vienna, Austria, 1115-1120, doi: 10.1109/cimca.2005.1631619.
[25] Chelly Dagdia, Z., Mirchev, M. (2020). When evolutionary computing meets astro- and geoinformatics, In: Škoda, P., Adam, F. (eds.), *Knowledge discovery in big data from astronomy and earth observation*, Elsevier, Amsterdam, Netherlands, 283-306, doi: 10.1016/B978-0-12-819154-5.00026-6.
[26] Wang, S., Liu, M. (2013). A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem, *Computers & Operations Research*, Vol. 40, No. 4, 1064-1075, doi: 10.1016/j.cor.2012.10.015.
[27] Valero-Gomez, A., Valero-Gomez, J., Castro-Gonzalez, A., Moreno, L. (2011). Use of genetic algorithms for target distribution and sequencing in multiple robot operations, In: *Proceedings of 2011 IEEE International Conference on Robotics and Biomimetics,* Karon Beach, Thailand, 2718-2724, doi: 10.1109/ROBIO.2011.6181716.
[28] Cbc: Version 2.9.9, from https://zenodo.org/record/1317566, accessed May 27, 2021.
[29] Bynum, M.L., Hackebeil, G.A., Hart, W.E., Laird, C.D., Nicholson, B.l., Siirola, J.D., Watson, J.P., Woodruff, D.L. (2021). *Pyomo - Optimization Modeling in Python*, Springer, Cham, Switzerland, doi: 10.1007/978-3-030-68928-5.
[30] Michalewicz, Z. (1999). The significance of the evaluation function in evolutionary algorithms, In: Davis, L.D., De Jong, K., Vose, M.D., Whitley, L.D. (eds.), *Evolutionary Algorithms, The IMA Volumes in Mathematics and its Applications*, Vol. 111, Springer, New York, USA, 151-166, doi: 10.1007/978-1-4612-1542-4_8.