An evolutionary approach for the offsetting inventory cycle problem

Chiara Franciosi*, Francesco Carrabs, Raffaele Cerulli and Salvatore Miranda

Abstract: In inventory management, a fundamental issue is the rational use of required space. Among the numerous techniques adopted, an important role is played by the determination of the replenishment cycle offsetting which minimizes the warehouse space within a considered time horizon. The NP-completeness of the Offsetting Inventory Cycle Problem (OICP) has led the researchers towards the development and the comparison of specific heuristics. We propose and implement a genetic algorithm for the OICP, whose effectiveness is validated by comparing its solutions with those found by a mixed integer programming model. The algorithm, tested on realistic instances, shows a high reduction of the maximum space and a more regular warehouse saturation with negligible increase of the total cost. This paper, unlike other papers currently available in literature, provides instances data and results necessary for reproducibility, aiming to become a benchmark for future comparisons with other OICP algorithms.

ABOUT THE AUTHORS

This work originates from the collaboration of the Department of Industrial Engineering and the Department of Mathematics of University of Salerno, Italy. Chiara Franciosi is a PhD student in Industrial Engineering at University of Salerno; her current research interests include Inventory Management, Industrial Maintenance Management, and Sustainable Manufacturing.

Francesco Carrabs is assistant professor at the Department of Mathematics of University of Salerno; his research interests include combinatorial optimization, heuristics, metaheuristics, hybrid heuristics for mixed integer linear programming problems, vehicle routing problems.

Raffaele Cerulli is Associate Professor at the Department of Mathematics of University of Salerno; his research interests include combinatorial optimization problems, mainly on mathematical models and algorithm design for covering problems on graphs and wireless sensor networks problems.

Salvatore Miranda is Associate Professor at the Department of Industrial Engineering of University of Salerno; his research interests include Operations Management, Maintenance, Human Reliability Analysis, Simulation, Logistics and Inventory Management.

PUBLIC INTEREST STATEMENT

One of the main issues in industrial companies concerns a proper inventory management and a suitable use of the warehouse space. Consequently, several techniques have been proposed to manage, in an efficient and effective way, the inventory and the limited warehouse space. The evolutionary approach proposed in this paper can be useful to manage the orders of a large amount of items/products with the aim of using in a better way the whole warehouse space, allowing larger order quantities while satisfying the market demand. In particular, the developed algorithm allows us to select the order quantities, which minimize ordering, holding and purchasing costs, and sets the optimal time of the first replenishment of each item in order to minimize the volume peak of the warehouse. Unlike other previous studies, this paper also provides data of instances and results of the algorithm applications to make possible future comparisons with other approaches.
1. Introduction

Inventory management is among the activities that show more criticality in an increasingly complex company environment. In the literature, many techniques have been investigated to optimize the use and management of the inventory in a warehouse. Among the most used models, the classic Wilson model allows minimization of the total cost through the use of an optimal lot for each item. The joint replenishment problem (JRP) of a family of items is based on the same assumptions as the Wilson model. It considers a uniform and deterministic demand for each item available in the storage, no shortages allowed, and no quantity discount. Consequently, cost savings could be achieved by coordinating the replenishment of \( N \) items. In the general JRP, since the quantities of each item are ordered for the first time at the initial instant of the considered period, the maximum occupied space occurs at the beginning of the period, and in the following days, the space will be used inefficiently with a resulting non-regular saturation of the warehouse. From this consideration, some heuristics have been developed with the aim of shifting the instant of first replenishment of the items and looking for the optimal offset that allows minimizing the maximum volume peak and guaranteeing a more regular saturation (e.g. Boctor, 2010; Murthy, Benton, & Rubin, 2003; Moon, Cha, & Kim, 2008; Yao & Chu, 2008). In this way, inefficiency can be avoided to a considerable extent, resulting in optimization of the inventory management process. This is a particularly complex problem, known in literature as Offsetting Inventory Cycle Problem (OICP) that cannot be solved optimally in polynomial time (Gallego, Shaw, & Simchi-Levi, 1992). Some heuristic approaches were implemented to obtain a good solution in a reasonable time. Anyway, to date, none of them have been validated and tested on instances of large dimensions showing the consequences of heuristic application in terms of warehouse space trend, value of maximum volume peak, and economic impact. In fact, although in some papers heuristics have been proposed and applied to small instances, none of them has detailed of data and results, therefore comparisons turn out to be impossible. This manuscript, according to the guidelines of Barr, Golden, Kelly, Resende, and Stewart (1995) for an appropriate representation of experimental results, provides all information necessary for reproducibility of the instances, aiming to become a benchmark for any future comparison. Furthermore, most of the papers regarding the OICP are based on mathematical assumptions (explained in detail at the end of the Literature Review Section) which, on one hand, reduce the problem complexity and, on the other hand, make the instances less realistic. Differently from what proposed up to now, this paper focuses on realistic instances data without any simplified assumption.

A heuristic procedure focused on the shifting of the first replenishment cycle of items in a warehouse and able to overcome the limits just highlighted, is designed, implemented, validated, and finally tested on new reproducible instances, proving its effectiveness through the achievement of a high minimization of the maximum peak during the considered time horizon and an optimal use of the space in the warehouse with a negligible increase of the total cost.

The paper, after a review of the state of the art related to the JRP and to the problem of offsetting inventory replenishment cycles, provides its detailed description and presents a genetic algorithm for its solution. Such algorithm is first validated with instances of small-medium dimension and then applied to two instances of more realistic dimensions showing its potentiality in terms of a more effective use of storage space.

2. Literature review

The JRP has been studied over 30 years. In 1989 Arkin proved that the JRP is an NP-hard problem; therefore, heuristics may be used for solving it, and many researches have addressed it (Cha, Moon,
& Park, 2008; Kaspi & Rosenblatt, 1991; Khouja, Michalewicz, & Satoskar, 2000; Lee & Yao, 2003; Tsai, Tsai, & Huang, 2009; Van Eijs, 1993; Viswanathan, 1996; Wang, He, Wu, & Zeng, 2012; Wildeman, Frenk, & Dekker, 1997). A review of the JRP literature up to the late 1980s was conducted by Aksoy and Selcuk Erenguc (1988) and Goyal and Satir (1989), while Khouja and Goyal (2008) present an interesting review of all the articles about this issue from 1989 to 2005.

Actually, there are many resource restrictions in inventory systems (e.g., space or budget), but the classic JRP does not consider this issue. For this reason, if a constraint is violated (e.g., the required space is greater than the available space), many textbooks suggest using Lagrange multipliers to find reduced order quantities that allow respecting the resource constraint (Hadley & Whitin, 1963; Johnson & Montgomery, 1974; Tersine, 1976). Many scientific papers have been written on this issue, in particular, the most recent are Amaya, Carvajal, and Castaño (2013), Haksever and Moussourakis (2005), Hoque (2006), Moon and Cha (2006), Yao (2007).

Another possible approach to the multi-item inventory system with resource constraints, particularly considered in the literature, assumes that all the items have a common and fixed cycle time (which represents the time between order (TBO) for each item). The most interesting papers in this frame are Chiu, Lin, Sheu, and Ho (2014), Haji and Mansuri (1995), Hall (1998), Rosenblatt and Rothblum (1990), Thomas and Hartley (1983), and Zoller (1977).

Other researchers focused their attention instead on the joint replenishment policy in which the cycle of each item is an integer multiple of an established basic cycle time (Goyal (1973); Kaspi and Rosenblatt (1983); Silver (1976)). More recently, Yao, Chu, and Lin (2008) studied this type of problem, suggested that the warehouse is supplied at the beginning of a basic period and proposed a new heuristics to generate a program for minimizing the maximum warehouse space, whereas Miranda, Fera, Iannone, and Riemma (2015) proposed a technique based on lot size modification in order to respect space limits with a reduced impact on the total warehousing cost.

However, the main problem of the JRP is that all the items are ordered at the beginning of the considered time horizon and, as a consequence, a maximum occupied space will occur at time \( t = 0 \) and will used inefficiently for the rest of the time. To solve this problem, as anticipated in the Introduction section, the offsetting can be carried out of N different items’ replenishment cycles that share a common space, and it is necessary to distribute the arrivals in an “intelligent” way in order to minimize the maximum peak in the warehouse, namely, the joint storage requirements for the items. This type of problem, which we are going to debate in this paper, is often called the “staggering problem” or “offsetting inventory cycle problem (OICP)”. In 1992, Gallego et al. showed that the staggering problem is NP-complete even if only one item has a different reorder interval and, thus, it is not possible to find the optimal solution in polynomial time, but a heuristic technique must be used.

Researches on the offsetting issue are rather few. Teo, Ou, and Tan (1998) deal with the OICP; however, the first real turning point came only in 2003 with the work of Murthy, Benton, and Rubin. Such authors considered the presence of space constraints and presented an interesting heuristics for offsetting independent and unrestricted ordering cycles for items in order to minimize their joint storage requirements over an infinite time horizon when warehouse space is limited. Given that the procedure developed by Murthy et al. represents the benchmark of many subsequent works in this field, from now on, we will call it the Murthy, Benton, and Rubin procedure (MBRP). Later, Moon et al. (2008) proposed an MIP model both for a finite and an infinite time horizon and a genetic algorithm and compared their procedures with the one previously presented by Murthy et al. (2003). Yao and Chu (2008) also conducted theoretical analysis based on Fourier series and Fourier transforms and proposed a procedure to calculate the maximum warehouse space requirement and showed the improvements compared with the MBRP. Then, Boctor (2010) proposed a new formulation of the MBRP and a heuristic algorithm based on simulated annealing through which they achieved the same results as Moon et al. (2008). Subsequently, Boctor and Bolduc (2012) presented two heuristic
approaches for solving the staggering problem and obtained good performances, while Croot and Huang (2013) studied this problem from the viewpoint of probability theory and proposed a series of algorithms for the OICP. Docteur and Bolduc (2015) presented two heuristic solution approaches to solve a bi-objective problem with the aim of minimizing the ordering and holding costs and the maximum required storage. In contrast, Franciosi, Miranda, Iannone, and Lambiase (2015) developed a first algorithm targeted to determine the optimal offsetting of item inventory cycles stored in the same warehouse. Finally, Russell and Urban (2016) proposed two heuristics for the OICP and they analysed new variants of the problem concerning a continuous-time framework as well as the effect of stochastic demand.

Most of the analysed papers consider the time horizon equal to the least common multiplier (LCM) of the times between orders (TBOs) associated to the items, since the maximum peak occurs during this time interval. In Yao et al. (2008) the authors proved that if this assumption holds then it is possible to fix the first replenishment instant of an item \( j \) at time zero without loss of generality. Thanks to this idea, it is possible to reduce the size of the problem, since all the variables associated to the item \( j \) are fixed a priori, and to reduce the symmetry issues that heavily affect the OICP.

Russell and Urban (2016) carried out some tests to verify the performance improvements obtained by fixing the first replenishment instant of one or more items. Moreover, they applied the different symmetry breaking parameters provided by CPLEX to state what is the best configuration to use for the OICP. For instance, on an instance with only 20 items CPLEX requires about 16 h to optimally solve it whereas, the same instance is solved in only 12 min by fixing the first replenishment instant of an item.

Due to the LCM assumption, the time horizon considered for the OICP is usually a finite value, for instance the 360 days of a year. In several papers concerning the OICP the authors select the TBOs as divisors of the time horizon assuring, in this way, that the first replenishment instant of, at least, one item can be fixed a priori. However, the choice of selecting the TBOs as divisors of the time horizon can be too much restrictive for companies that need to define the TBO of each item with the aim of reducing the stock management cost. For this reason, in this paper the time horizon is fixed to 220 days, corresponding to average number of annual working days, and the TBOs are chosen within the range 1–220 without restrictions. Consequently, TBOs may not be divisors of 220 and then the LCM assumption does not hold in our instances and no first replenishment instants can be fixed a priori. This choice makes the problem more realistic but even more complex and it represents a significant novelty of this paper compared to the existing literature. Moreover, since the daily demand is equal to the ratio between the replenishment quantity and the TBO, its value can be fractional as well as the peak value.

Such considerations together with the necessity to provide data and results of the instances, have led the authors to propose a heuristic able to solve realistic and reproducible cases.

3. Problem description and formulation

The goal of the OICP is to find the optimal offsetting of each item with the aim of decreasing the maximum volume peak in the warehouse and, consequently, stabilizing the saturation of the storage for better management of space.

The Wilson model is widely used for supply problems, and it determines the optimal ordering lot for each item, known as EOQ, which minimizes the sum of the purchasing, ordering, and holding costs. Furthermore, for simplification, this model fixes the first replenishment time for each item to the initial instant of the time horizon, and thus, the items will be available simultaneously in the warehouse with a consequent remarkable peak in volume at the initial time and a considerably unsaturated warehouse in the following days.

The example below better explains the problem just described.
Consider the data reported in Table 1: storage space, daily demand, TBOs, and EOQ for each item.

If it considers a time horizon fixed to 20 days, the space required separately from the three items is shown in the Figure 1, indicated by black dotted lines, while the total space requirement without offsetting, namely, with the application of the Wilson model, is shown by a red line. It can be seen that the maximum peak occurs at the initial instant of the time horizon and is equal to 59 m³.

When the offsetting is carried out, the items are ordered for the first time on days zero, six, and two, as shown in the Figure 2, and not simultaneously at the beginning of the time horizon. Consequently, the initial quantities are set equal to 9, 12, and 10, respectively. As can be seen, the maximum peak does not occur at the initial instant of the time horizon; it is lower than the previous one and is equal to 49 m³. In this manner, it is possible to have better management of space.

Table 1. Data of the example

| Item | Storage space per unit time (m³) | Daily demand | Time between orders - TBO (days) | Economic order quantity - EOQ |
|------|---------------------------------|--------------|----------------------------------|-------------------------------|
| #1   | 1                               | 3            | 3                                | 9                             |
| #2   | 1                               | 2            | 10                               | 20                            |
| #3   | 1                               | 5            | 6                                | 30                            |

Figure 1. Space requirement without offsetting.

Figure 2. Space requirement with offsetting.
In this simple example, there are only three items in the storage, but in the real cases, the number of items involved significantly increases; therefore, the aforementioned problem is much more remarkable. Consequently, there are two main criticalities to face in the OICP:

(1) Violation of potential space constraints.
(2) Bad management of the warehouse due to nonhomogeneous saturation of space.

In the present work, the same assumptions of Franciosi et al. (2015), Moon et al. (2008), and Murthy et al. (2003) about the OICP are used: the daily demand is deterministic and constant, the replenishment is instantaneous, the TBO is known and constant for each item over a finite time horizon, and shortages and backlogs are not allowed. The objective is to minimize the maximum peak during the considered time horizon. Since the total space requirement pattern is periodic, the maximum peak will occur in the time interval from \( t = 0 \) to \( t = \text{LCM}(\text{TBO}_1, ..., \text{TBO}_N) \), where LCM is the least common multiple and corresponds to the whole time horizon. However, in real cases that involve a large number of items, the LCM of the TBOs may become very high, leading to an overblown time horizon. For this reason, we use a finite time horizon \( T \) that makes the problem more realistic, as also explained by Franciosi et al. (2015). In our case, \( T \) is fixed to 220 days, which approximately correspond to a working year.

To each item \( j \), three parameters are associated: demand \( d_j \), time between orders \( \text{TBO}_j \), and replenishment quantity \( Q_j = d_j \cdot \text{TBO}_j \). The formulation of the OICP is based on two sets of variables: \( x_{jt} \) and \( I_{jt} \).

The binary variable \( x_{jt} \) is equal to 1 if and only if an order for item \( j \) occurs at time \( t \), with \( t = 0, 1, ..., \text{TBO}_j - 1 \). The variable \( I_{jt} \) represents the inventory level for item \( j \) at time \( t \), with \( t = 0, 1, ..., T \). Finally, we introduce a variable \( S_{\text{max}} \) equal to the maximum storage space required for all items in the time interval \( t \in [0, T] \). Since the \( d_j \) values can be not integers, \( I_{jt} \) and \( S_{\text{max}} \) are continuous variables.

\[
\text{(MIP) } \min S_{\text{max}} \tag{1}
\]

subject to:

\[
\sum_{t=0}^{\text{TBO}_j-1} x_{jt} = 1 \quad j = 1, ..., N \tag{2}
\]

\[
I_{j0} = Q_j x_{j0} + d_j \sum_{t=1}^{\text{TBO}_j-1} t x_{jt} \quad j = 1, ..., N \tag{3}
\]

\[
I_{jt} = I_{j(t-1)} + Q_j x_{jt} - d_j \quad j = 1, ..., N; \quad t = 1, ..., \text{TBO}_j - 1 \tag{4}
\]

\[
S_{\text{max}} \geq \sum_{j=1}^{N} s_j \cdot k(j) = \begin{cases} 
    t & \text{if } t < \text{TBO}_j \\
    \mod\left(t/\text{TBO}_j\right) & \text{if } t \geq \text{TBO}_j 
\end{cases} \quad t = 0, ..., T \tag{5}
\]

\[
x_{jt} \in \{0, 1\} \quad j = 1, ..., N; \quad t = 0, ..., \text{TBO}_j - 1 \tag{6}
\]

\[
I_{jt} \geq 0 \quad j = 1, ..., N; \quad t = 0, ..., \text{TBO}_j - 1 \tag{7}
\]

\[
S_{\text{max}} \geq 0 \tag{8}
\]

The objective function (1) minimizes the maximum space required in the warehouse over the considered time horizon \( T \). Constraints (2) force the first replenishment instant of any item \( j \) to occur
within the range \([0, TBO−1]\). Constraints (3) and (4) ensure that the value of variables \(I_j\) coincides with the inventory level of item \(j\) at time \(t\). More in details, due to the constraints (2) we know that there is exactly one \(x_{j,t}\) equal to 1 into the constraints (3). Now, if \(x_{j,0} = 1\) then the inventory level of the item \(j\) is equal to \(Q_j\) because the replenishment instant for the item \(j\) is zero. Otherwise, if \(x_{j,0} = 1\) then \(I_{j,0}\) is equal to \(t\) times the daily demand \(d_j\). About the constraint (4), if \(x_{j,t} = 0\) then \(I_j\) is equal to the inventory level of item \(j\) at time \(t−1\) minus the daily demand \(d_j\). Otherwise, if \(x_{j,t} = 1\) then \(I_j\) is given by replenishment quantity \(Q_j\) minus the daily demand \(d_j\). Finally, for any instant of time \(t\), constraints (5) force the variable \(S_{\text{max}}\) to be greater than or equal to the sum of the inventory level of all items.

The MIP model will be used to verify the effectiveness of the genetic algorithm, described in the next section. This comparison will show that the genetic algorithm is able to find good solutions even for large instances that involve many items in a warehouse.

4. Genetic algorithm

Genetic algorithms are bioinspired metaheuristic techniques introduced by Holland (1975) in his book *Adaptation in Natural and Artificial Systems*. These techniques, based on natural selection and evolution, reproduce the evolutionary process of the species. The genetic algorithms consider a population of chromosomes (or individuals) that represent feasible or unfeasible solutions to the problem. The quality of an individual, namely, how the solution is good for the problem, is measured by a fitness function created ad hoc for the specific problem. Therefore, a genetic algorithm is an iterative search procedure whose purpose is optimizing the fitness function. Starting from an initial population, normally generated randomly, a genetic algorithm produces new generations usually containing better individuals than the previous ones: the algorithm progresses to the global optimum of the fitness function.

The great ability of a GA to explore in depth the solutions space and the possibility to effectively manage the constraints through the setting of few parameters, make the genetic algorithm, among evolutionary algorithms, potentially suitable for the OICP.

It is necessary to appropriately set the parameters of the genetic algorithm to obtain good solutions. The possible parameters are known thanks to the many articles and books presented in the literature: Aytug, Khouja, and Vergara (2003), Dowsland (1996), Hua and Huang (2006), Li and Gen (1996), Maiti, Bhunia, and Maiti (2006), Mitchell (1998), and Yokota, Gen, Li, and Kim (1996) are examples of guidelines for GA configuration and setting.

In the OICP, the variables (individuals of the genetic algorithm) are restricted to integer values.

The logic of our genetic algorithm is explained by the flowchart in Figure 3 and through the pseudo-code in Figure 4. The algorithm randomly assumes (for the first generation, \(z = 1\)) the values of the first replenishment instant for each item, and in subsequent generations, it researches, through genetic reproduction, the values of \(g_{j,z}\) that lead to the minimization of the maximum peak \(S_{\text{max}}\).

Starting from the first item \((j = 1)\), initial instant of time \((t = 0)\), and first generation \((z = 1)\), the genetic algorithm randomly assigns the first replenishment instant for the first item \((g_{1,1})\), and if \(g_{1,1} = 0\), the initial quantities \(I_{1,0,1} = Q_1\); otherwise \(I_{1,0,1} = g_{1,1} \cdot d_1\). The values of \(I_{1,0,1}\) are memorized. The procedure is repeated for each item \(j\), and then the total storage space required for all items is calculated according to the equation

\[
S_{0,z} = \sum_{j=1}^{N} S_j \cdot I_{j,0,z}
\]  

From the following day until the last day of the time horizon \(T\) is reached, the algorithm calculates the present quantities in the warehouse according to the equation
Figure 3. Flowchart of the genetic algorithm.

Figure 4. Pseudo-code of the genetic algorithm.

Read: N, TBO, sj, Qj, dj, T
z = 1
FOR z = 1 to Z
  t = 0
  j = 1
  FOR j = 1 to N
    IF z = 1 THEN Assign randomly (gj, z)
    ELSE Genetic Reproduction (gj, z)
    END IF
    IF gj, z = 0 THEN Ij, t, z = Qj
    ELSE Ij, 0, z := gj, z + dj
    END IF
    Memorize Ij, 0, z
    j = j + 1
  NEXT j
  S0,z = \sum_{j=1}^{N} sj \cdot Ij, 0, z
  Memorize St, z
  t = t + 1
  FOR t = 1 to T
    FOR j = 1 to N
      Ij, t, z := Ij, t-1, z - dj
      IF Ij, t, z < dj THEN Ij, t, z := Ij, t, z + Qj
      END IF
      j = j + 1
    NEXT j
    St, z := \sum_{t=1}^{T} sj \cdot Ij, t, z
    Memorize St, z
    t = t + 1
  NEXT t
  Smax, z = \max \{ St, z \}
  Memorize Smax, z, gj, z
  z = z + 1,
  NEXT z
  Sbest := min Smax, z
  Write: Sbest, Best
\[ I_{j,t,z} = I_{j,t-1,z} + Q_j - d_j \]  
\[ \text{in which } Q_j = 0 \text{ if } I_{j,t-1,z} \geq d_j, \text{ while } Q_j = d_j \cdot \text{TBO}_j \text{ if } I_{j,t-1,z} < d_j. \] 
Moreover, the total space \( S_{t,z} \) occupied by the items every day is calculated. At the end of this iterative procedure, the algorithm computes the maximum space occupied by the items during the time horizon \( T \) and memorizes this value, according to the equation:

\[ S_{max} = \max_t \{ S_{t,z} \}. \]

The aforementioned procedure is repeated for all generations considering the genetic reproduction that leads to the best values of the first replenishment instants \( (t_{best}, j) \) and the optimal value of the maximum space required by the items in the warehouse \( (S_{best}) \). When the maximum number of generation \( Z \) is reached, the algorithm memorizes and shows the best value of the maximum space required by items in the warehouse \( (S_{best}) \) and the corresponding best first replenishment instant for every item \( (t_{best}, j) \) that led to the value \( S_{best} \). Generally, as the algorithm is structured, \( S_{best} \) coincides with \( S_{max} \) of the last generation.

In the following paragraphs, we present the choices made for each parameter of the genetic algorithm.

### 4.1. Chromosome representation and initial population

In our genetic algorithm, the length of each chromosome (individual) is equal to the number of items within the warehouse; accordingly, each chromosome has as many genes as the items considered and each gene \( j \) represents the first replenishment instant \( g_{j,z} \) of the corresponding item \( j \). Therefore, it is necessary that the gene is an integer between 0 (if \( g_{j,z} = 0 \), it means that the lot will be ordered for the first time at the initial instant as in the Wilson model) and \( TBO_j - 1 \) for each item. If we consider the example of three items presented in Section 3, the individual representation of the solution shown in Figure 2 is the following:

\[ \begin{align*} 
0 & \quad 6 & \quad 2 
\end{align*} \]

The initial population, namely the first set of solutions, is created randomly, and the population size is fixed equal to 50 individuals after a tuning phase carried out with a range of individuals between 20 and 200 in several instances and with different number of items. By increasing the population size over 50 individuals, we observed a negligible improvement of the solutions found with a significant increase in the computational time.

### 4.2. Fitness function

The fitness function evaluates the goodness in the individuals of the population. In our problem, the fitness of each chromosome is equal to the maximum peak obtained according to the value of each gene. In every generation, the algorithm will choose individuals with the best fitness value for the following generation that minimizes the space according to Equation (1).

### 4.3. Methods for selecting individuals

The chosen technique used for the selection of parents is the tournament selection, which involves running several tournaments among individuals chosen at random from the population; then, the individual with the best fitness value, namely the winner of the tournament, is selected for the following crossover. The tournament size is set equal to three. The tournament selection is explained and studied in several previous articles such as Blickle and Thiele (1995), Goldberg and Deb (1991), Miller and Goldberg (1995), which affirmed that tournament selection is a useful, simple, and robust selection mechanism commonly used in GAs. Moreover, Goldberg and Deb (1991) showed that tournament selection has better or equivalent convergence and computational time complexity properties when compared to any other reproduction operator that exists in the literature. Finally, as clearly
explained by Deb (2000), to manage constraints the tournament selection is combined with a penalty function, which allows easily selecting only feasible individuals for next generations.

4.4. Operators to vary genetic composition of individuals during the reproduction: Crossover and mutation

The individuals that survive to the selection step undergo a change through the application of the crossover and mutation operators with the aim of generating new individuals in the next generation.

In our case, “Laplace crossover” and “power mutation” have been chosen as reproduction operators. Laplace crossover is a parent centric real coded operator based on Laplace distribution and it was introduced by Deep and Thakur (2007a), while power mutation is an operator based on power distribution described by Deep and Thakur (2007b). Deep and Thakur (2007a, 2007b) tested Laplace crossover and power mutation operators on several algorithms and on 20 benchmark problems available in global optimization literature showing that the GA using jointly such operators emerged as the best. Moreover, these operators integrate a truncation procedure for integer restrictions (Deep, Singh, Kansal, & Mohan, 2009), necessary in our GA for getting integer variables. According to Deep et al. (2009), the crossover rate and the mutation rate are respectively fixed to 0.8 and 0.005.

4.5. Stopping criteria

The established stopping condition is a maximum number of generations. In fact, as indicated in the detailed review conducted by Aytug et al. (2003) concerning the use of genetic algorithms to solve production and operations-management problems, the most common criterion used for stopping a genetic algorithm is a fixed number of generations. After a series of simulations with a large variety of instances, the number of generations, which more often permits the convergence to the optimal solution or very close to the optimal solution/upper bound of MIP model, is fixed to 300. In fact, a number of generations equal to 300 is the right compromise between obtained results and acceptable run times: when the number of generations increases, the run time consequently increases, while the result remains almost constant.

5. Validation

With the aim of testing the validity and the effectiveness of the two aforementioned procedures, both have been applied to the example presented in the literature by Murthy et al. (2003). The data of the Murthy et al. example (occupied space and TBO for each item), which consider nine items, are shown below in Table 2.

As previously mentioned, the fixed time horizon is equal to a working year ($T = 220$ days).

The Wilson model was applied to this example and, as can be seen in the following table, the highest volume peak without offsetting is equal to 1035 m$^3$. It occurs at the beginning of the considered time period because all the lots $Q_j$ are ordered simultaneously at the initial instant of the time horizon ($g_j = 0, \forall j$). Murthy et al., through their procedure, obtained a maximum volume peak equal to 875 m$^3$ with a reduction of 15.46%. Instead, with the application of the offsetting through the MIP model and the genetic algorithm, we are able to obtain a maximum peak equal to the optimum value of 760 m$^3$, with a percentage reduction of the occupied volume equal to 26.57%.

| Table 2. Data of Murthy et al.’s example |
|-----------------------------------------|
| Item $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $s_jQ_j$ | 100 | 200 | 81 | 144 | 150 | 160 | 90 | 60 | 50 |
| TBO, $j$ | 4 | 5 | 9 | 12 | 15 | 8 | 6 | 12 | 2 |
As shown in Table 3, the maximum peak values found by genetic algorithm and MIP model coincide and this means that the genetic algorithm solves optimality this example. Therefore, in Table 3, the first replenishment instant for each item ($g_j$) and the corresponding space occupied at the initial instant for each item ($I_{j,0} \cdot s_j$), which leads to the maximum volume peak equal to 760 m$^3$, are reported.

Figure 5 shows the space requirement without offsetting and with the application of the genetic algorithm during the considered time horizon $T$: the black line represents the occupied space with the application of the classic Wilson model without offsetting; the red line, instead, represents the space requirement after the application of the genetic algorithm.

A reduction of the storage space is possible and a more regular saturation of the warehouse permits a more correct management of the items and the space.

The next section shows the results obtained with the application of the MIP model and the genetic algorithm to eight new instances of different dimension, whose data are reported in the Appendix A.

**Table 3. Application of the two procedures to Murthy et al.’s example**

| Item | Without offsetting | Murthy et al. model | MIP model | Genetic algorithm |
|------|--------------------|---------------------|-----------|------------------|
|      | $g_j$  | $I_{j,0} \cdot s_j$ | $g_j$  | $I_{j,0} \cdot s_j$ | $g_j$  | $I_{j,0} \cdot s_j$ |
| #1   | 0      | 100                | 0       | 100              | 3      | 100               |
| #2   | 0      | 200                | 4       | 160              | 80     | 1                 |
| #3   | 0      | 81                 | 0       | 81               | 5      | 54                |
| #4   | 0      | 144                | 0       | 144              | 1      | 24                |
| #5   | 0      | 150                | 0       | 150              | 3      | 40                |
| #6   | 0      | 160                | 6       | 120              | 5      | 120               |
| #7   | 0      | 90                 | 5       | 75               | 4      | 75                |
| #8   | 0      | 60                 | 4       | 20               | 7      | 40                |
| #9   | 0      | 50                 | 1       | 25               | 0      | 25                |
|      | Max peak (m$^3$) | 1,035             | 875      | 760              | 760    |
| Peak day | 0  | 0  | 38  | 42  |
| % Reduction | - | 15.46% | 26.57% | 26.57% |

Notes: The values in bold are representative of the results obtained through the application of MIP model and genetic algorithm on the Murthy et al.’s example.
6. Experimental tests

The purpose of the computational tests presented in this section is to study the effectiveness and the performance of our genetic algorithm. Both the genetic algorithm and the MIP model have been applied to several instances with 10, 20, 30, 40, 50, 80, and 200 items, respectively.

Table 4 reports the range of values in which are included the main characteristics of the considered items in all cases: the daily demand, the TBOs, and, consequently, the economic order quantity. For simplification, the specific volume is fixed to 1 m³/unit.

Table 5 reports the results obtained by the application of the MIP model to the different instances.

It has to be noted that, since the peak value can be a not integer value, the upper bounds reported in the following table are rounded up to the closest integer value.

The MIP model was coded in C++ on an OSX platform running on an Intel i5 2.9 GHz processor with 16 GB of RAM and solved by using the Concert library of IBM ILOG CPLEX 12.5. We fixed a limit of 3 h and 8 GB of memory for the resolution of each instance. In each row of the table, the upper and lower bounds computed by the MIP model for that instance, the computational time, the stop status, and the gap between the upper and lower bound are reported. Obviously, when an optimal solution is found, the upper and lower bounds coincide and the gap is equal to 0%.

The instances with 9 and 10 items are optimally solved by the MIP model, while it reaches the time limit on the instances with 20 items, 30 items, 40 items, 50 items, 80 items and 200 items. The gap values are low and the upper bounds found by the MIP model are close enough to the optimal solution.

Table 6 shows the solution values found by MIP model and genetic algorithm in each case study. Moreover, the run times of the genetic algorithm and the percent gap between the maximum peaks of the two procedures are reported.

| Table 4. Characteristics of the items |
|--------------------------------------|
| Characteristic | Measure   | Unit  | Range          |
| Daily demand (d) | Units/day | 1–106 |
| Time between order (TBO) | Days  | 1–220 |
| Economic order quantity (Q) | Units   | 74–3,748 |

| Table 5. MIP model results |
|----------------------------|
| Case study | Upper bound (m³) | Lower bound (m³) | Run time (sec) | Stop motivation | GAP (%) |
|------------|------------------|------------------|----------------|----------------|---------|
| 9 items    | 760.00           | 760.00           | 3.55           | Optimal        | 0.00    |
| 10 items   | 2,854.00         | 2,854.00         | 229.95         | Optimal        | 0.00    |
| 20 items   | 7,121.00         | 6,902.06         | 10,810.01      | AbortTimeLim   | 3.07    |
| 30 items   | 12,352.00        | 12,210.00        | 10,810.03      | AbortTimeLim   | 1.14    |
| 40 items   | 13,795.00        | 13,585.30        | 10,810.03      | AbortTimeLim   | 1.52    |
| 50 items   | 17,602.00        | 17,131.80        | 10,810.04      | AbortTimeLim   | 2.67    |
| 80 items   | 27,689.00        | 27,024.60        | 10,810.08      | AbortTimeLim   | 2.40    |
| 200 items  | 73,309.00        | 72,477.70        | 10,810.18      | AbortTimeLim   | 1.13    |
The genetic algorithm is able to obtain the optimum value only for the instance with 9 items, while, in the other cases, it finds solutions very close to the upper bounds of the MIP model with a percent error always lower than 3.3%. Regarding performance, the genetic algorithm produces results very fast, requiring less than 5 min to solve all the instances up to 80 items and 12 min for the instance with 200 items. Since the running time is so low, we tested the algorithm even on larger instances, which involve, respectively, 1,000 and 2,000 items within the boundaries of a company warehouse. The results of these tests are described in the next section.

6.1. Larger instances (1,000 and 2,000 items)
The genetic algorithm was applied to two more realistic cases of companies managing, respectively, 1,000 and 2,000 items in the warehouse. Table 7 reports the ranges in which the main data of the instances are included, namely, daily demand, TBOs, specific volume, unit purchasing cost, unit ordering cost, and unit holding cost for each case.

Once the data was collected, the economic order quantity $Q_j$ has been calculated and, considering the classic Wilson model that established the presence of all quantities for the first time at $t = 0$, it obtains a maximum volume peak equal to 318,488 m$^3$ for the case of 1,000 and 645,813.24 m$^3$ for the case of 2,000 items. Both peaks occur on the initial instant of the first day.

With the application of the proposed genetic algorithm, the maximum volume peak is equal to 175,813 m$^3$ for the case of 1,000 items, which occurs on day 217 with a reduction of occupied space of approximately 44.8%. In the case of 2,000 items, the genetic algorithm obtains a maximum peak equal to 355,386.67 m$^3$, which occurs on the day 46, with a reduction of the occupied space of approximately 45%.

Figures 6 and 7 show the respective space requirements for the two cases during the considered time horizon $T$: the black line represents the occupied space with the application of the classic Wilson model without offsetting; the red line represents the space requirement after the application of the genetic algorithm.

### Table 6. Comparison between the results of the two procedures

| Case study | MIP model | Genetic Algorithm |
|------------|-----------|-------------------|
|            | Max Volume Peak (m$^3$) | Max Volume Peak (m$^3$) | Run Time (s) | Gap (%) |
| 9 items    | 760       | 760               | 87           | 0.000   |
| 10 items   | 2,854.00  | 2,881.00          | 88           | 0.946   |
| 20 items   | 7,121.00  | 7,229.00          | 110          | 1.517   |
| 30 items   | 12,352.00 | 12,660.00         | 137          | 2.494   |
| 40 items   | 13,795.00 | 14,147.00         | 177          | 2.552   |
| 50 items   | 17,602.00 | 18,019.00         | 206          | 2.369   |
| 80 items   | 27,689.00 | 28,489.00         | 294          | 2.889   |
| 200 items  | 73,309.00 | 75,682.00         | 733          | 3.237   |

### Table 7. Ranges of values for instances

| Characteristic     | Measure unit | Range 1,000 items | Range 2,000 items |
|--------------------|--------------|-------------------|-------------------|
| Daily demand (d)   | Units/day    | 1–108             | 1–110             |
| Time between order (TBO) | Days       | 1–202             | 1–373             |
| Specific volume (s) | m$^3$/unit   | 0.002–1           | 0.002–1           |
| Unit purchasing cost (p) | €/unit     | 0.1–7             | 0.1–7             |
| Unit ordering cost (Cl) | €/order    | 50–100            | 50–100            |
| Unit holding cost (k)  | €/(unit-day) | 0.001–0.05       | 0.001–0.05        |
In both cases, a huge difference between the two procedures regarding the occupied volume, the saturation of the warehouse, can be observed. Moreover, with the application of offsetting through the genetic algorithm, better management of the warehouse is possible, and a major space, usable for other items or for potential safety stocks, is made available.

Furthermore, in real cases it is fundamental to consider the costs. Table 8 presents the total purchasing, ordering, and holding costs associated with 1,000 items and 2,000 items and incurred by the company with both the Wilson model and the genetic algorithm.

As evidenced in Table 8, the purchasing cost remains constant in the two cases because such cost does not depend on the specific lot, but only on the daily demand, the time horizon, and the unit purchasing price.
Instead, the ordering and holding costs depend on the specific lot. For this reason, with the application of the genetic algorithm, it is necessary to consider an extra ordering cost and an extra holding cost for the hypothesized initial quantities that must satisfy the demand of the first days until the first replenishment fixed by the genetic algorithm occurs. However, the considered time horizon is always the same and, for this reason, the ordering and holding costs (associated to the lot $Q_j$) with the genetic algorithm are inferior to the Wilson model because in the initial days only the initial quantities with the associated extra ordering and holding costs have to be considered. After the first replenishment and until the end of the time horizon, the ordering and holding costs are the same associated to the lot $Q_j$, and so they are calculated in the same way for the two models.

In both cases, the total cost with the application of the genetic algorithm is greater, but the difference, about 0.8% for both cases, is negligible if compared with the value of the total ordering and holding cost calculated for each case and with the obtained reduction of the occupied space.

### 7. Conclusions
The present paper deals with the OICP. A genetic algorithm, whose features have been accurately chosen for this problem, has been introduced and then validated through comparison with the solutions computed by an MIP model. The data and results of all instances used for the validation are reported in order to make possible their reproducibility and future comparisons in the OICP field.

The algorithm has shown good performance also on realistic cases, which involved a large number of items and without any simplified mathematical assumption. The obtained solutions reveal that a huge reduction in the space requirement is possible and that a more regular saturation of the warehouse allows for a better use of space. Therefore, the algorithm is suitable for any type of instance and it is able to handle real cases in effective way.

Further research could be focused on testing different heuristic techniques, applying them to real cases, and comparing the results. Another possible next step would concern a model modification by including in it the space constraint imposed by the finite dimension of the warehouse, as usually happens in many logistic problems. In this last case, the objective function won’t be the occupied space minimization but the minimization of the total cost of stock management with the respect of the new space constraint.

### Table 8. Comparing the costs

| Characteristic          | 1,000 item instance | 2,000 item instance |
|-------------------------|---------------------|---------------------|
|                         | Wilson model | Genetic algorithm | Wilson model | Genetic Algorithm |
| Purchasing cost (€)     | 38,375,050  | 38,375,050  | 75,632,639  | 75,632,639  |
| Ordering cost (€)       | 1,373,354   | 1,336,834   | 2,766,932   | 2,693,515   |
| Holding cost (€)        | 1,371,553   | 1,335,040   | 2,769,247   | 2,695,727   |
| Extra ordering cost (€) | –          | 71,325      | –            | 145,386      |
| Extra holding cost (€)  | –          | 22,865      | –            | 46,664       |
| Total cost (€)          | 41,119,957  | 41,141,114  | 81,168,818  | 81,213,731  |
| Max volume (m³)         | 318,488     | 175,813     | 645,813.24  | 355,386.67  |
Notations

The following notations were used in this paper:

- $j$: index for each item
- $N$: number of items
- $T$: finite time horizon
- $z$: index for each genetic algorithm generation
- $Z$: maximum number of genetic algorithm generations
- $x_j$: binary variable that indicates, for each item, when an order occurs
- $g_j$: integer variable that indicates the first replenishment instant for each item $j$
- $g_{j,z}$: integer variable that indicates the first replenishment instant for each item $j$ at each generation $z$
- $d_j$: daily demand for the $j$th item
- $TBO_j$: time between orders for the $j$th item
- $Q_j$: replenishment quantity for the $j$th item
- $I_{j,t}$: quantity of the $j$th item present in the warehouse at time $t$
- $I_{j,t,z}$: quantity of the $j$th item present in the warehouse at time $t$ and at generation $z$
- $s_j$: space required per unit time for the $j$th item
- $S_{t,z}$: total storage space required for all items at time $t$ and at generation $z$
- $S_{max}$: maximum storage space required for all items in the finite time horizon
- $t_{best,j}$: best first replenishment instant for the $j$th item

Funding

The authors received no direct funding for this research.

Author details

Chiara Franciosi
E-mail: cfranciosi@unisa.it
ORCID ID: http://orcid.org/0000-0002-9983-1386

Francesco Carrabs
E-mail: fcarrabs@unisa.it
Raffaele Cerulli
E-mail: raffaele@unisa.it
Salvatore Miranda
E-mail: smiranda@unisa.it

1 Department of Industrial Engineering, University of Salerno, via Giovanni Paolo II, 132, Fisciano, SA 84084, Italy.
2 Department of Mathematics, University of Salerno, via Giovanni Paolo II, 132, Fisciano, SA 84084, Italy.

Citation information

Cite this article as: An evolutionary approach for the offsetting inventory cycle problem, Chiara Franciosi, Francesco Carrabs, Raffaele Cerulli & Salvatore Miranda, Cogent Engineering (2017), 4: 1370764.

References

Aksoy, Y., & Selcuk Erenec, S. (1988). Multi-item inventory models with co-ordinated replenishments: A survey. International Journal of Operations & Production Management, 8, 63–73. https://doi.org/10.1108/eb054814

Amaya, C. A., Carvajal, J., & Castaño, F. (2013). A heuristic framework based on linear programming to solve the constrained joint replenishment problem (C-JRP). International Journal of Production Economics, 144, 243–247. https://doi.org/10.1016/j.ijpe.2013.02.008

Arkin, E., Joneja, D., & Roundy, R. (1989). Computational complexity of uncapacitated multi-echelon production planning problems. Operations Research Letters, 8, 61–66. https://doi.org/10.1016/0167-6377(89)90001-1

Aytug, H., Khouja, M., & Vergara, F. E. (2003). Use of genetic algorithms to solve production and operations management problems: A review. International Journal of Production Research, 41, 3955–4009. https://doi.org/10.1080/00207540310001626319

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G., & Stewart, W. R., Jr. (1995). Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics, 1, 9–32. https://doi.org/10.1007/BF02430363

Blickle, T., & Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. Technical Report 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich.

Boctor, F. F. (2010). Offseting inventory replenishment cycles to minimize storage space. European Journal of Operational Research, 203, 321–325. https://doi.org/10.1016/j.ejor.2009.07.035

Boctor, F. F., & Bolduc, M. C. (2012). The inventory replenishment planning and staggering problem revisited. Montreal: CIRRELT.
Boctor, F. F., & Belduc, M. C. (2015). Inventory replenishment planning and staggering. IFAC-PapersOnLine, 48, 1416–1421. https://doi.org/10.1016/j.ifacol.2015.06.285

Cha, B. C., Moon, I. K., & Park, J. H. (2008). The joint replenishment and delivery scheduling of the one-warehouse, n-retailer system. Transportation Research Part E: Logistics and Transportation Review, 44, 720–730. https://doi.org/10.1016/j.tre.2007.05.010

Chiu, C. Y., Lin, Y., Sheu, D. F., & Ho, W. T. (2014). Common replenishment cycle with mixed batch shipment policy for a single-vendor multi-buyer integrated system. European Journal of Industrial Engineering, 8, 168–192. https://doi.org/10.1504/EJIE.2014.060435

Croo, E., & Huang, K. (2013). A class of random algorithms for inventory cycle offsetting. International Journal of Operational Research, 18, 201–217. https://doi.org/10.1504/IJOR.2013.056115

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186, 311–338. https://doi.org/10.1016/S0045-7825(99)00389-8

Deep, K., Singh, K. P., Kansal, M. L., & Mohan, C. (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems. Applied Mathematics and Computation, 212, 505–518. https://doi.org/10.1016/j.amc.2009.02.044

Deep, K., & Thakur, M. (2007a). A new crossover operator for real coded genetic algorithms. Applied Mathematics and Computation, 188, 895–911. https://doi.org/10.1016/j.amc.2006.10.047

Deep, K., & Thakur, M. (2007b). A new mutation operator for real coded genetic algorithms. Applied Mathematics and Computation, 193, 211–230. https://doi.org/10.1016/j.amc.2007.03.046

Dowsland, K. A. (1996). Genetic algorithms—a tool for OR? Journal of the Operational Research Society, 47, 550–561. https://doi.org/10.1057/jors.1996.60

Franciosi, C., Miranda, S., Iannone, R., & Lambiase, A. (2015, September 21–23). A heuristics with simulative approach for the determination of the optimal offsetting replenishment cycles to reduce the warehouse space. In The 14th International Conference on Modeling and Applied Simulation, MAS 2015 (pp. 17–25), Bergeggi.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms, 1, 69–93.

Goyal, S. K. (1975). Adaption in natural and artificial systems. Ann Arbor, MI: The University of Michigan Press.

Hoque, M. A. (2006). An optimal solution technique for the joint replenishment problem with storage and transport capacities and budget constraints. European Journal of Operational Research, 175, 1013–1042. https://doi.org/10.1016/j.ejor.2005.06.022

Hua, Z., & Huang, F. (2006). An effective genetic algorithm approach to large scale mixed integer programming problems. Applied Mathematics and Computation, 174, 897–909. https://doi.org/10.1016/j.amc.2005.05.017

Johnson, L. A., & Montgomery, D. C. (1974). Operations research in production planning, scheduling, and inventory control (Vol. 6). New York, NY: Wiley.

Kaspi, M., & Rosenblatt, M. J. (1983). An improvement of Silver’s algorithm for the joint replenishment problem. AIIE Transactions, 15, 264–267.

Kaspi, M., & Rosenblatt, M. J. (1991). On the economic ordering quantity for jointly replenished items. International Journal of Production Research, 29, 107–114. https://doi.org/10.1080/00207549108930051

Khouja, M., & Goyal, S. (2008). A review of the joint replenishment problem literature. 1989–2005. European Journal of Operational Research, 186(1), 1–16. https://doi.org/10.1016/j.ejor.2007.03.007

Khouja, M., Michalewicz, Z., & Sotoskar, S. S. (2000). A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. Production Planning & Control, 11, 556–564. https://doi.org/10.1080/095372800414115

Lee, F. C., & Yao, M. J. (2003). A global optimum search algorithm for the joint replenishment problem under power-of-two policy. Computers & Operations Research, 30, 1319–1333. https://doi.org/10.1016/S0305-0548(02)00073-4

Li, Y. X., & Gen, M. (1996). Nonlinear mixed integer programming problems using genetic algorithm and penalty function. IEEE International Conference on Systems, Man, and Cybernetics, 4, 2677–2682.

Maiti, A. K., Bhunia, A. K., & Maiti, M. (2006). An application of real-coded genetic algorithm (RCGA) for mixed integer non-linear programming in two-storage multi-item inventory model with discount policy. Applied Mathematics and Computation, 183, 903–915. https://doi.org/10.1016/j.amc.2006.05.141

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 9, 193–212.

Miranda, S., Fera, M., Jannone, R., & Riemma, S. (2015). A multi-item constrained EOQ calculation algorithm with exit condition: A comparative analysis. IFAC-PapersOnLine, 48, 1314–1319. https://doi.org/10.1016/j.ifacol.2015.06.267

Mitchell, M. (1998). An introduction to genetic algorithms. Cambridge, MA: MIT press.

Moon, I. K., & Cho, B. C. (2006). The joint replenishment problem with resource restriction. European Journal of Operational Research, 173, 190–198. https://doi.org/10.1016/j.ejor.2004.11.020

Moon, I. K., Cha, B. C., & Kim, S. K. (2008). Offsetting inventory cycles using mixed integer programming and genetic algorithm. International Journal of Industrial Engineering: Theory, Applications and Practice, 15, 245–256.

Murthy, N. N., Benton, W. C., & Rubin, P. A. (2003). Offsetting inventory cycles of items sharing storage. European Journal of Operational Research, 150, 304–319. https://doi.org/10.1016/S0377-2217(02)00518-0

Rosenblatt, M. J., & Rothblum, U. G. (1990). On the single resource capacity problem for multi-item inventory systems. Operations Research, 38, 686–693. https://doi.org/10.1287/opre.38.4.686
Russell, R. A., & Urban, T. L. (2016). Offsetting inventory replenishment cycles. European Journal of Operational Research, 254, 105–112. https://doi.org/10.1016/j.ejor.2016.03.055

Silver, E. A. (1976). A simple method of determining order quantities in joint replenishments under deterministic demand. Management Science, 22, 1351–1361. https://doi.org/10.1287/mnsc.22.12.1351

Teo, C. P., Ou, J., & Tan, K. C. (1998). Multi-item inventory staggering problems: Heuristics and bounds. In Proceedings of the Ninth Annual Ace-Siam Symposium on Discrete Algorithms, By Association for Computing Machinery. San Francisco, CA.

Tersine, R. J. (1976). Materials management and inventory systems. Amsterdam: North Holland.

Thomas, L. C., & Hartley, R. (1983). An algorithm for limited capacity inventory problem with staggering. Journal of the Operational Research Society, 34, 81–85. https://doi.org/10.1057/jors.1983.10

Tsai, C. Y., Tsai, C. Y., & Huang, P. W. (2009). An association clustering algorithm for can-order policies in the joint replenishment problem. International Journal of Production Economics, 117, 30–41. https://doi.org/10.1016/j.ijpe.2008.08.056

Van Eijs, M. J. G. (1993). A note on the joint replenishment problem under constant demand. Journal of the Operational Research Society, 44, 185–191. https://doi.org/10.1057/jors.1993.31

Viswanathan, S. (1996). A new optimal algorithm for the joint replenishment problem. Journal of the Operational Research Society, 47, 936–944. https://doi.org/10.1057/jors.1996.119

Wang, L., He, J., Wu, D., & Zeng, Y. R. (2012). A novel differential evolution algorithm for joint replenishment problem under interdependence and its application. International Journal of Production Economics, 135, 190–198. https://doi.org/10.1016/j.ijpe.2011.06.015

Wildeman, R. E., Frenk, J. B. G., & Dekker, R. (1997). An efficient optimal solution method for the joint replenishment problem. European Journal of Operational Research, 99, 433–444. https://doi.org/10.1016/S0377-2217(96)00072-0

Yoo, M. J. (2007). Solving the joint replenishment problem with warehouse-space restrictions using a genetic algorithm. Journal of the Chinese Institute of Industrial Engineers, 24, 128–141. https://doi.org/10.1080/10170660709509028

Yoo, M. J., & Chu, W. M. (2008). A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements. Omega, 36, 619–631. https://doi.org/10.1016/j.omega.2007.01.007

Yoo, M. J., Chu, W. M., & Lin, Y. F. (2008). Determination of replenishment dates for restricted-storage, static demand, cyclic replenishment schedule. Computers & Operations Research, 35, 3230–3242. https://doi.org/10.1016/j.cor.2007.02.020

Yokota, T., Gen, M., Li, Y., & Kim, C. E. (1996). A genetic algorithm for interval nonlinear integer programming problem. Computers & Industrial Engineering, 31, 913–917. https://doi.org/10.1016/0360-8352(96)00263-X

Zoller, K. (1977). Deterministic multi-item inventory systems with limited capacity. Management Science, 24, 451–455. https://doi.org/10.1287/mnsc.24.4.451
## Appendix A

| #Item | $Q_{10}$ |
|-------|----------|
| 1–10  | 617 677 81 193 238 341 692 219 524 603 |

| #Item | $TBO_{10}$ |
|-------|------------|
| 1–10  | 6 20 81 32 30 23 115 17 8 10 |

| #Item | $Q_{20}$ |
|-------|----------|
| 1–10  | 113 823 741 423 562 442 274 1,043 443 803 |
| 11–20 | 612 1,343 594 565 406 426 412 508 508 596 |

| #Item | $TBO_{20}$ |
|-------|------------|
| 1–10  | 23 39 9 11 6 8 14 33 44 38 |
| 11–20 | 10 14 14 14 9 7 9 9 7 7 |

| #Item | $Q_{30}$ |
|-------|----------|
| 1–10  | 544 326 370 726 547 249 160 1,470 1,316 607 |
| 11–20 | 618 544 583 972 1,145 485 986 200 335 520 |
| 21–30 | 2,961 420 178 756 686 410 1,086 441 505 189 |

| #Item | $TBO_{30}$ |
|-------|------------|
| 1–10  | 7 8 10 10 13 83 40 22 14 9 |
| 11–20 | 7 8 10 13 19 6 10 25 19 10 |
| 21–30 | 51 16 15 10 8 7 33 5 12 38 |

| #Item | $Q_{40}$ |
|-------|----------|
| 1–10  | 261 521 643 408 1,169 192 592 696 396 143 |
| 11–20 | 535 514 706 303 692 501 1,510 614 330 459 |
| 21–30 | 331 74 108 354 263 422 654 327 548 1,378 |
| 31–40 | 862 792 962 197 304 1,424 442 636 2,023 345 |

| #Item | $TBO_{40}$ |
|-------|------------|
| 1–10  | 11 5 10 19 32 19 13 10 11 20 |
| 11–20 | 7 9 9 14 10 8 21 11 13 9 |
| 21–30 | 19 74 36 13 10 11 9 8 91 15 |
| 31–40 | 10 24 13 25 10 24 25 6 38 10 |
| #Item | Q_{50} | TBO_{50} | Q_{80} | TBO_{80} |
|-------|--------|----------|--------|----------|
|       |        |          |        |          |
| 1–10  | 502    | 108      | 652    | 844      |
| 11–20 | 1,647  | 465      | 857    | 865      |
| 21–30 | 1,021  | 150      | 2,424  | 989      |
| 31–40 | 127    | 735      | 1,298  | 385      |
| 41–50 | 130    | 297      | 850    | 679      |
|       |        |          |        |          |
| 1–10  | 63     | 27       | 7      | 11       |
| 11–20 | 36     | 13       | 15     | 12       |
| 21–30 | 13     | 21       | 25     | 19       |
| 31–40 | 42     | 7        | 16     | 9        |
| 41–50 | 33     | 15       | 12     | 10       |
|       |        |          |        |          |
| 1–10  | 576    | 1,269    | 644    | 560      |
| 11–20 | 525    | 503      | 1,115  | 475      |
| 21–30 | 192    | 482      | 524    | 581      |
| 31–40 | 435    | 539      | 329    | 331      |
| 41–50 | 466    | 485      | 1,010  | 453      |
| 51–60 | 583    | 914      | 755    | 339      |
| 61–70 | 354    | 778      | 683    | 174      |
| 71–80 | 1,533  | 484      | 566    | 1,604    |
|       |        |          |        |          |
| 1–10  | 9      | 14       | 8      | 11       |
| 11–20 | 12     | 42       | 31     | 8        |
| 21–30 | 19     | 14       | 7      | 13       |
| 31–40 | 12     | 6        | 10     | 9        |
| 41–50 | 33     | 5        | 11     | 7        |
| 51–60 | 8      | 10       | 31     | 10       |
| 61–70 | 14     | 9        | 11     | 58       |
| 71–80 | 26     | 6        | 13     | 25       |
| #Item | $Q_{200}$ | $TBO_{200}$ |
|-------|-----------|-------------|
| 1–10 | 518, 769, 808, 450, 322, 730, 725, 739, 415, 603 | 10, 17, 135, 10, 7, 8, 9, 18, 9, 10 |
| 11–20 | 508, 353, 894, 266, 3,140, 1,348, 552, 138, 415, 324 | 6, 10, 10, 10, 10, 34, 17, 15, 23, 13, 20 |
| 21–30 | 105, 543, 388, 1,068, 710, 289, 579, 288, 1,030, 758 | 35, 9, 15, 33, 15, 13, 6, 17, 13, 10 |
| 31–40 | 473, 238, 622, 646, 732, 820, 529, 449, 2,923, 1,106 | 9, 22, 11, 9, 8, 12, 9, 9, 29, 17 |
| 41–50 | 461, 491, 1,017, 517, 599, 569, 192, 371, 692, 422 | 6, 5, 20, 7, 12, 11, 21, 13, 9, 10 |
| 51–60 | 299, 1,013, 1,033, 1,532, 1,538, 501, 1,519, 1,704, 507, 810 | 13, 18, 13, 26, 17, 10, 17, 34, 9, 8 |
| 61–70 | 1,015, 660, 326, 1,143, 427, 610, 559, 690, 364, 295 | 11, 9, 82, 52, 5, 7, 7, 33, 11, 10 |
| 71–80 | 191, 1,095, 424, 782, 315, 606, 537, 520, 1,511, 628 | 27, 17, 6, 17, 8, 17, 23, 20, 16, 37 |
| 81–90 | 150, 622, 644, 546, 573, 775, 101, 484, 414, 184 | 25, 16, 19, 12, 15, 12, 51, 9, 10, 20 |
| 91–100 | 320, 494, 361, 342, 363, 558, 157, 631, 634, 1,120 | 11, 7, 9, 14, 11, 8, 26, 10, 14, 14 |
| 101–110 | 2,258, 466, 174, 1,387, 416, 496, 444, 519, 395, 525 | 6, 5, 20, 7, 12, 11, 21, 13, 9, 10 |
| 111–120 | 1,375, 392, 3,748, 1,061, 456, 549, 160, 289, 519, 91 | 121–130, 251, 597, 236, 1,895, 558, 983, 544, 416, 2,265, 709 |
| 121–130 | 369, 746, 554, 770, 315, 474, 681, 729, 247, 646 | 151–160, 831, 190, 743, 151, 932, 400, 515, 369, 345, 351 |
| 131–140 | 405, 468, 449, 417, 124, 595, 707, 251, 472, 1,044 | 161–170, 566, 661, 582, 1,063, 360, 840, 777, 519, 718, 386 |
| 141–150 | 1,015, 660, 326, 1,143, 427, 610, 559, 690, 364, 295 | 171–180, 681, 306, 646, 391, 618, 679, 589, 1,471, 85, 562 |
| 151–160 | 534, 588, 561, 1,982, 647, 475, 1,524, 594, 355, 915 | 181–190, 31, 7, 6, 20, 9, 6, 54, 13, 6, 2 |
| 161–170 | 251, 597, 236, 1,895, 558, 983, 544, 416, 2,265, 709 | 191–200, 31, 7, 6, 20, 9, 6, 54, 13, 6, 2 |
