

Integration of a Heuristic Method into an ERP Software: A Successful Experience for a Dynamic Multi-item Lot Sizing Problem

José M. Gutiérrez, Marcos Colebrook, Cristian Rivero, and Teresa Pestana

1 Introduction

Nowadays, logistics costs (among those are the inventory carrying and ordering costs) as a percentage of gross domestic product are about 10 % in OECD countries [1]. However, these logistics expenditures in Latin America and Caribbean, as a percentage of the final product value, are between 18 and 35 % higher than those of OECD countries [2]. Accordingly, in the current economic scenario, where the different operations in the supply chain are globalized, it is imperative that at a microeconomic level small and medium enterprises, or SMEs for short, define efficient strategies that make them more competitive with rivals. One of these approaches consists of determining either optimal or near-optimal ordering policies, which significantly reduce the inventory and replenishment costs. Precisely, Inventory Management, as a field of Operations Research, provides an appropriate methodology to deal with these challenges from a scientific prism.

Moreover, we are interested in designing a solution method to determine a joint replenishment plan for multiple items that are to be stored at either a wholesaler/

J.M. Gutiérrez (✉)

Departamento de Matemáticas, Estadística e Investigación Operativa, Universidad de La Laguna, Tenerife, Spain

e-mail: jmgtrrez@ull.edu.es

M. Colebrook

Departamento de Ingeniería Informática, Universidad de La Laguna, Tenerife, Spain

e-mail: mcolesan@ull.edu.es

C. Rivero

Urban Science, Madrid, Spain

e-mail: crivero@urbanscience.com

T. Pestana

ITOP MC, Tenerife, Spain

e-mail: teresapestana@itop.es

retailer, when items are finished goods, or an in-process storage area in case of raw materials, work-in-process goods, components or spare parts. Therefore, we assume that a capacitated warehouse/depot within the supply chain is replenished from external suppliers and/or internal manufacturing processes to thereafter satisfy the deterministic time-varying demands for multiple items of end-customers, retailers or ERP (*Enterprise Resource Planning*), which is an information system that integrates and automates key business processes within the company.

Thus, we consider that each time an order for an item is placed a setup cost (fixed and independent of the order quantity) and a linear replenishment cost are incurred. Additionally, a carrying cost dependent on the ending inventory level is applied for each item and period. Furthermore, the demand for any item and period must be fulfilled on time and instantly, i.e. stockouts are not permitted and leadtimes are negligible. The problem can be mathematically described using Mixed Integer Programming (MIP) and solved by any commercial optimization software. However, since the number of items can be huge in practice, even the most powerful optimization program would consume too much time in determining an optimal solution. Accordingly, ad-hoc heuristic methods become interesting tools to obtain compromise solutions.

The first contribution to determine an optimal replenishment plan under deterministic time-varying parameters is credited to Wagner and Whitin [3]. It is a fact that the model considered by the authors is extremely idealistic. In particular, a single-item is produced/replenished in an uncapacitated single-facility and the demand varies with time through a finite planning horizon and should be met without shortages in each period. Furthermore, the cost structure consists of the sum of linear carrying cost and fixed production/replenishment cost (regardless of the order quantity) for all periods. However, a key result was derived in this paper, which has been extensively used afterward in other more complex dynamic models. A family of optimal solutions called Zero Inventory Ordering (ZIO) policies were introduced for the first time. These policies define a simple decision rule: a production/replenishment order is placed only when the inventory level is zero. Thereafter, Veinott [4] and Zangwill [5] proved that this rule can be also applied even when the cost structure is concave in general.

In the early seventies, Love [6] proposed an extension of the Wagner-Whitin model to include both limited storage capacities and general concave cost structures for inventory carrying, placing orders and backlogging. Moreover, a polynomial solution method based on the Dynamic Programming (DP) paradigm was derived to construct optimal plans. Although the theoretical complexity of the algorithm by Love was not improved, Gutiérrez et al. [7, 8] introduced a new characterization of the optimal plans that allows devising a DP algorithm with faster running times and linear expected time complexity.

Unfortunately, references to the model with multiple items are quite sparse in the literature. Possibly, this lack of interest is due to research efforts have focused primarily on the Capacitated Lot Sizing Problems (CLSP, family of problems where the constraints are imposed on the production capacity rather than storage limitations). One reason for this disinterest could be that researchers have

traditionally considered that what applies to the CLSP also applies to the other model. However, this approach is not entirely true since whilst the CLSP is NP-Complete in general [9], the Wagner-Whitin model with storage capacities is not. Conversely, the latter model is as complex as the CLSP when multiple products are involved, and hence heuristics become attractive methods to generate good approximate solutions thereby offsetting the high running times consumed by commercial solvers.

At the end of the last decade, Minner [10] carried out a comprehensive study in which three different heuristic strategies were implemented and compared. These approaches are based on different methodologies. On the one hand, the *smoothing approach* by Dixon and Poh [11] consists of determining independently the replenishment policy for each item first by using any of the efficient algorithms proposed for the original dynamic lot sizing problem (namely, [12–14]). In a second step, infeasibilities in one period are fixed either postponing to the next period (PUSH) or advancing to the previous period (PULL) the replenishment quantity that minimizes the marginal cost. Secondly, the *constructive approach* devised by Günther [15] determines first lot-for-lot policies to solely satisfy the demand for each product in each period. After that, the replenishment quantities for each item and period are increased using an economic criterion as a discriminant. Finally, the *savings approach* developed by Axsäter [16] for the vehicle routing problem consists of reducing the ordering costs by combining consecutive replenishments in one batch whenever this operation represents a saving. The computational experiment in Minner [10] suggests that the last method is more robust than the other two heuristics when demand variability increases; and shows smaller increases of deviations for variations of costs and capacity parameters.

The rest of this chapter is organized as follows. In Sect. 2 we formulate the model in terms of a Mixed Integer Programming (MIP) problem. The key results to improve the approach proposed by Dixon and Poh [11] and the heuristic method are described in detail in Sect. 3. Moreover, the integration of the heuristic method as an add-on in the commercial software SAP Business One is explained in Sect. 4. In Sect. 5, we introduce a visual example to illustrate both how the input parameters should be entered and how the final solution is reported to the decision maker. Finally, in Sect. 6 we provide some conclusions and final remarks.

2 Problem Formulation

We assume a set of N items independently demanded and a planning horizon with T periods. Besides, we define the following parameters (see Table 1):

Moreover, we denote by $D_{n,t}$ the accumulated demand of item n from period t to T , namely $D_{n,t} = \sum_{k=t}^T d_{n,k}$. Additionally, let S_t be the total dynamic inventory capacity at the warehouse in period t and let w_n be the unit capacity (volume) of item n . On the other hand, we define the following variables (see Table 2):

Table 1 The set of parameters

Parameter	Description
$d_{n,t}$	Demand for item n in period t
$f_{n,t}$	Fixed setup cost for item n in period t
$p_{n,t}$	Replenishment cost for item n in period t
$h_{n,t}$	Carrying cost for item n in period t

Table 2 The set of variables

Variable	Description
$x_{n,t}$	Order quantity replenished at the beginning of the period t for item n
$y_{n,t}$	Binary variable related to the order quantity, which is equal to 1 if an order for item n is placed in period t , and 0 otherwise
$I_{n,t}$	Inventory level of item n at the end of period t

Moreover, we assume that both the initial and final inventory level for each item is zero, i.e. $I_{n,0} = I_{n,T} = 0$ for all items n . However, these assumptions can be dropped off without loss of generality since the case of positive initial and/or final inventory can be adapted to the formulation below just allocating initial inventories to demands of the first periods and/or adding a required final inventory to the demand of the last period. Accordingly, we can state the following MIP problem called the *Multi-Item Ordering/Production Problem with Storage Constraints* (MIOPPSC):

$$\begin{aligned}
 & \text{(MIOPPSC)} \min \sum_{n=1}^N \sum_{t=1}^T f_{n,t} y_{n,t} + p_{n,t} x_{n,t} + h_{n,t} I_{n,t} \\
 & \text{s.t. :} \\
 & \sum_{n=1}^N w_n (I_{n,t-1} + x_{n,t}) \leq S_t, t = 1, \dots, T \\
 & I_{n,t-1} - I_{n,t} + x_{n,t} = d_{n,t}, t = 1, \dots, T; n = 1, \dots, N \\
 & x_{n,t} \leq y_{n,t} D_{n,t}, t = 1, \dots, T; n = 1, \dots, N \\
 & x_{n,t}, I_{n,t} \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}; y_{n,t} \in \{0, 1\}, t = 1, \dots, T; n = 1, \dots, N
 \end{aligned}$$

The terms in the objective function represent, respectively, the total setup cost, the total ordering cost and the total holding cost. The first set of constraints ensures that the inventory level at the beginning of each period does not exceed the warehouse capacity. The next constraint set are the well-known material balance equations and the third set states the relationship between the order quantity and its binary variable for each item and period. Finally, the last set of constraints defines the character of each variable.

3 The Heuristic Approach

Inspired by the smoothing approach by Dixon and Poh [11], we have proposed in Gutiérrez et al. [17] a heuristic method to determine near-optimal ordering schedules for a set of items, which share a common warehouse within a demand system not necessarily independent (horizontal). In case of dependent demand systems, we confine ourselves to consider only the final requirements for those items (raw material, sub-assemblies, intermediate assemblies and sub-components) sharing the same depot/container after the Bill of Materials (BOM) explosion is accomplished. The new method is based on the natural extension of the characterization of the optimal plans for the single-item case in Gutiérrez et al. [7] to consider multiple items.

As in Dixon and Poh [11], the first step of our heuristic consists of determining independent plans by solving N single-product, dynamic uncapacitated lot-sizing problems. If infeasibility occurs at the end of period t , the excess warehouse capacity requirements can be reduced either by postponing an item replenishment batch (PUSH operation) or by advancing a future replenishment (PULL operation). Nevertheless, Dixon and Poh claimed that on average the PUSH operation is preferred to the PULL strategy since the impact of the move on total costs is predictable. Thus, our heuristic differs from that given by Dixon and Poh in that we confine ourselves to analyze the PUSH operation but extending the strategy to consider both postponements of orders for a subset of items to nonconsecutive periods (i.e., from period t to period $t+k$, $k \in \{1, \dots, T-t\}$, instead of $t+1$ only) and replenishment quantities different from a sum of demands of consecutive periods for an item.

Precisely, let $\hat{x}_{n,t}$ denote the optimal replenishment quantity in period t of the independent plan for item n , then when the exceeded capacity $W_t = \sum_{n=1}^N w_n(I_{n,t-1} + \hat{x}_{n,t}) - S_t$ is positive in period t , the replenishment quantity in period t for a selection of items must be partially postponed to later periods. The items are chosen from those minimizing the marginal cost of shifting one unit of item n from a period $i \leq t$ to a later period $j > t$. Obviously, the items must be selected assuring the feasibility at period t , hence the total storage capacity should be not exceeded and the total replenishment must satisfy at least the demand of each item in that period.

For the sake of clarity, let us assume that the storage capacity constraint at period t is violated ($W_t > 0$). Moreover, let us admit that it is convenient shifting forward from period i to period j a certain replenishment quantity of item n . Note that although the infeasibility occurs at period t , we do not confine ourselves to consider only those items that were replenished at that period in the corresponding independent optimal plans. Instead, we also determine the marginal costs $c_{i,j}^n$ for those items n replenished in a period i prior to t . Accordingly, let $\text{succ}(n, t)$ and $\text{pred}(n, t)$ denote the production periods successor and predecessor, respectively, to period t in the independent optimal plan for item n . It should be noticed that period j is strictly less

than the production period $succ(n, t)$ successor to t in the independent optimal plan for item n . Consequently, period j was not a production period in that plan and so the corresponding setup cost must be included to determine the actual cost of the postponement, which is defined as:

$$r_{i,j}^n = f_{n,j} + c_{i,j}^n q_{t,j}^n$$

where $q_{t,j}^n$ represents the actual quantity that is postponed from period i to period j for item n . Observe that $q_{t,j}^n$ can be either a sum of demands of consecutive periods for item n or a quantity enough to remove the exceeded inventory in period t . However, in order to make period t feasible, withdrawing a sum of demands for a single item could not be sufficient, and hence we should select a subset of items N_t such that the sums of demands $\sum_{k \in N_t} q_{t,j_k}^k$ with minimum marginal costs are equal to or greater than W_t . Consequently, for each unfeasible period, we consider two types of sorted lists, namely, a list $L[n]$ for each item n , which contains increasing marginal costs corresponding to sums of demands of consecutive periods and a unique list G including the marginal costs of those items for which W_t is a multiple of their unit capacities.

Once these lists are completed, we should compare the sum of accumulated marginal costs (CLW) for a selection of items obtained from lists $L[n]$ with the minimum marginal cost (CW) in G . Remark that W_t should be updated each time the replenishment of one item is partially shifted to a subsequent period. If after the postponement the storage capacity remains to be violated, then we must select a different item to reduce the excess warehouse capacity requirements. We show below a sketch of the heuristic method to determine near-optimal solutions.

Algorithm MIOPPSC

For the sake of clarity, we denote the data structures as: $W(t)$, $w(n)$, etc.

for all N items **do**

Solve each uncapacitated plan independently using
Wagelmans et al. (1992) algorithm

end for

for each period t where the capacity $S(t)$ is violated in the previous plans

for all items $n \leftarrow 1$ to N **do**

Compute list G and lists $L[n]$

end for

$CW \leftarrow$ minimum cost value of list G

$CLW \leftarrow 0$

while $W(t) > 0$ **do**

$CLW \leftarrow CLW +$ minimum cost of all lists $L[n]$, $n = 1$ to N

if $CLW < CW$ **then**

```

    Decrease  $W(t)$  by the corresponding quantity  $q$  multiplied by
    the item's weight  $w(n)$ 
    Recalculate lists  $G$  and  $L[n]$ 
else
     $W(t) \leftarrow 0$ 
end if
end while
    Update properly plan  $x$  and all subsequent variables
end for

```

The reader is referred to Gutiérrez et al. [17] for a comprehensive description of both the algorithm and the computational experiment, which reports that the solutions provided by the heuristic are on average 5 % above the best solution achieved by CPLEX. Furthermore, the heuristic has been afterward encapsulated into an ERP. In this sense, Iris and Yenisey [18] recently addressed a very similar model to the one presented in this chapter, and they plan to embed it also into an ERP system.

4 Development and Implementation Within SAP Business One

SAP Business One is an ERP developed by SAP that is focused on small and medium enterprises (SME), and that covers various fields: accounting, procurement, sales, inventory management, production and manufacturing, customer relation management (CRM), etc.

The development process along with the subsequent deployment was carried out as a joint collaboration with ITOP Management Consulting, a company created in 2006 in Tenerife (Spain), that offers its services on information technology primarily to SME. ITOP holds a certification as an authorized SAP partner, as well as an UNE 166002 certification on R&D Management granted by AENOR (Spanish Association for Standardization and Certification), being one of the first national companies in obtaining this type of certification.

The algorithm described in a previous section that solves the MIOPPSC problem was programmed in C# using Microsoft Visual Studio [19] and the SAP Business One SDK (Software Development Kit) [20] as an *add-on*. These SAP Business One add-ons are basically programs developed to extend the functionality in order to satisfy the final user requirements.

The main components of the system are:

- SAP Business One Server: it holds the database that contains all the company's data such as information about the business partners, products, invoices, etc.
- SAP Business One Clients: the MIOPPSC add-on is installed on them, and they connect to the server to get the required information.

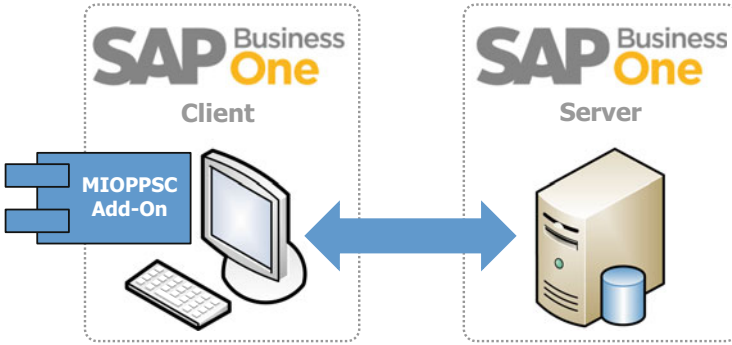


Fig. 1 System architecture integration between the MIOPPSC add-on and SAP Business One

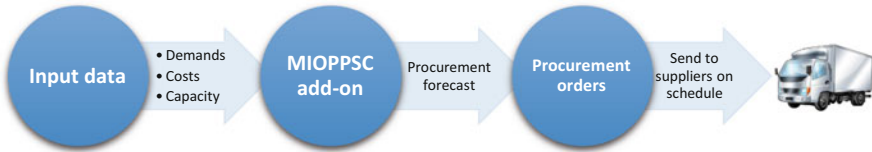


Fig. 2 Data flow from the initial input to the final output

The following Fig. 1 summarizes the integration between SAP Business One and the add-on.

The flow of information inside SAP Business One from the initial input to the final solution is as follows (see Fig. 2):

1. The algorithm takes, for each period $t = 1, \dots, T$, the input data for the demands $d_{n,t}$, the setup costs $f_{n,t}$, and the inventory capacity of the warehouse S_t .
2. This input data is then used by the MIOPPSC algorithm to compute a procurement forecast.
3. Finally, this forecast is used by the MRP (*Material Requirements Planning*) module to determine the procurement orders to send to the suppliers in order to satisfy the customers' demands over each period.

To illustrate this workflow, we show in the next section a visual example running on a real case.

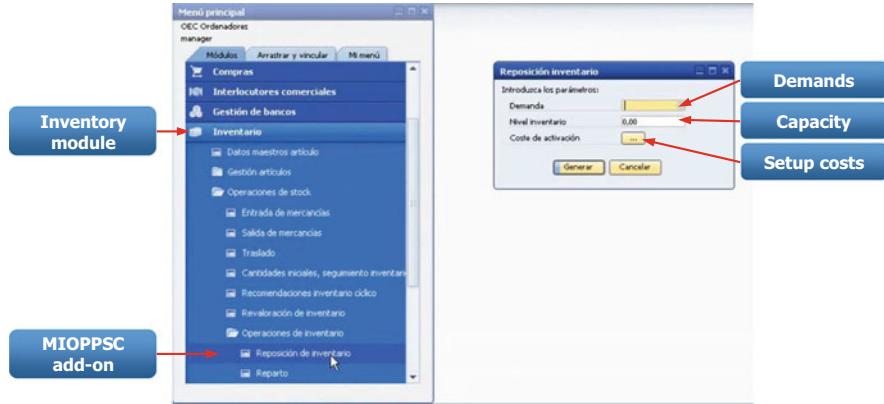


Fig. 3 Input parameters of the MIOPPSC add-on running within SAP Business One

5 Visual Example

First of all and, in order to show the example, we need to run the MIOPPSC add-on that was installed under the *Inventory module* of SAP Business One (see Fig. 3).

The input parameters needed by MIOPPSC are explained in the following sections.

5.1 Demands

The demands for each item will be taken from a pre-defined object in SAP Business One called *forecasts* within the MRP module, which are the known demands in future periods (see Fig. 4).

In this particular case, the demands for the two items were identical:

$$d_{1,t} = d_{2,t} = \{69, 29, 36, 61, 61, 26, 34, 67, 45, 67, 79, 56\}, \quad t = 1, \dots, 12$$

The forecast panel has several parts:

- *Forecast information*: such as the name and the description.
- *Forecast dates*: initial and final dates.
- *Month view*: in our example, the periods are expressed in months within a year range (from April to March which corresponds to $t = 1, \dots, 12$).
- *Item information*: such as item code and its description. In this example, we only have two items ($n = 1, 2$).
- *Item demands*: for each period (months).

The rest of these two items' parameters ($n = 1, 2$), such as the unit capacity (volume, w_n), and both the production costs (p_t) and the holding/carrying costs (h_t),

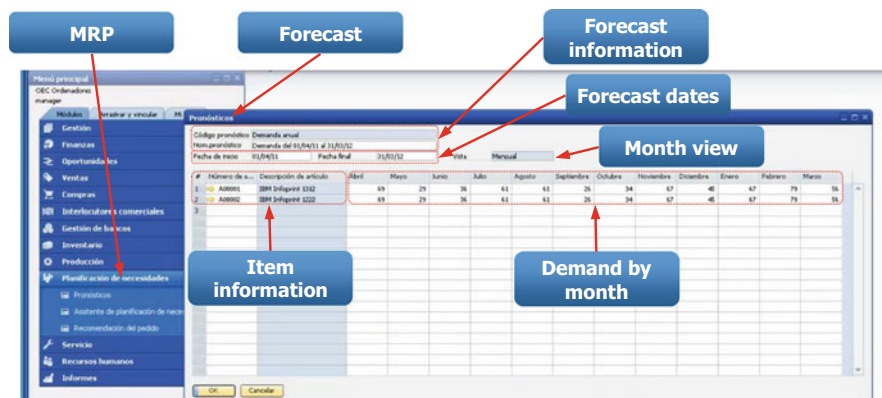


Fig. 4 Demand forecast of two items for each period (month) as the input to the add-on

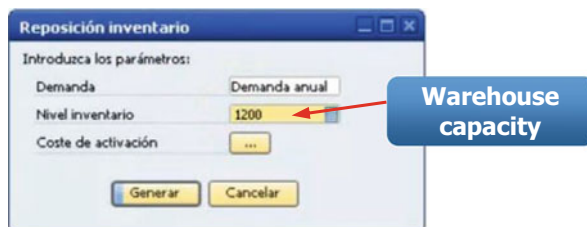


Fig. 5 The capacity (inventory level) of the warehouse is set for all the periods

with $t = 1, \dots, 12$, can be taken from each item's master data. In this particular example, the values of these parameters are the following:

- Unit capacity: $w_1 = 5$, $w_2 = 10$.
- Production costs: $p_{1,t} = p_{2,t} = 0$, for all periods $t = 1, \dots, 12$.
- Holding/carrying costs: $h_{1,t} = h_{2,t} = 1$, for all periods $t = 1, \dots, 12$.

5.2 Capacity

In this example, the capacity (inventory level of the warehouse) will be set to 1,200 for all periods $t = 1, \dots, 12$ (see Fig. 5).

5.3 Setup Costs

For each period $t = 1, \dots, 12$, the setup costs of each item must be entered manually as we show in Fig. 6. The values are the following:

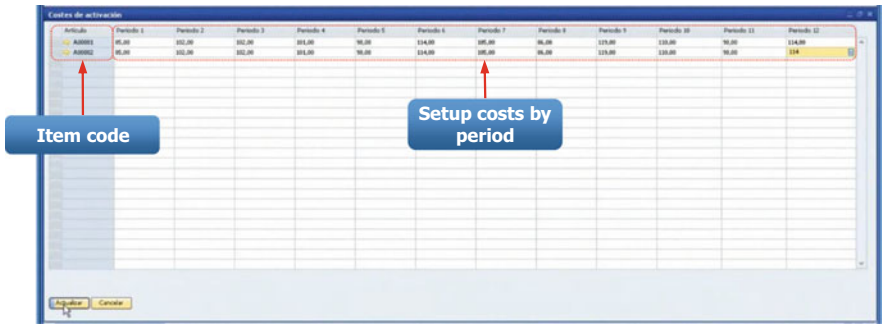


Fig. 6 Setup costs for the two items and the twelve periods considered in the example



Fig. 7 Example ready to be solved

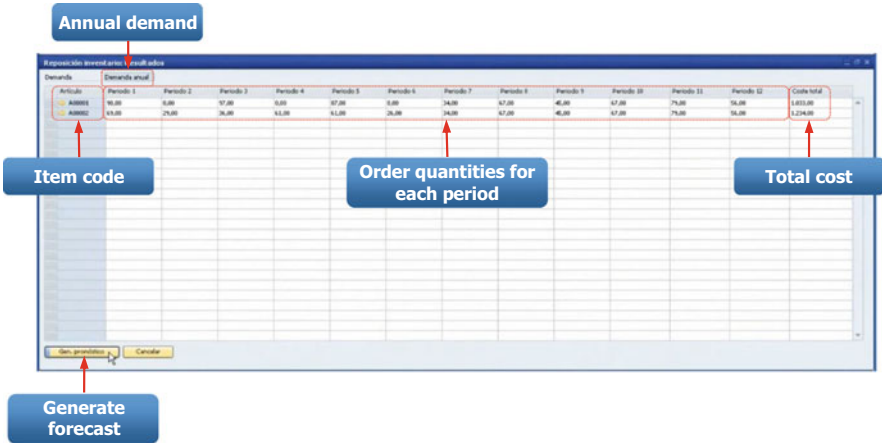


Fig. 8 Solution to the example showing the order quantities to be replenished at the beginning of each period

$$f_{1,t} = f_{2,t} = \{85, 102, 102, 101, 98, 114, 105, 86, 119, 110, 98, 114\}$$

Once all these data is set up, we can press the corresponding button to solve the problem (see Fig. 7).

The solution output will point out the order quantities of each item ($n = 1, 2$) to be replenished at the beginning of the corresponding period (months, $t = 1, \dots, 12$), in order to minimize the total cost (see Fig. 8). The solution values are the following:

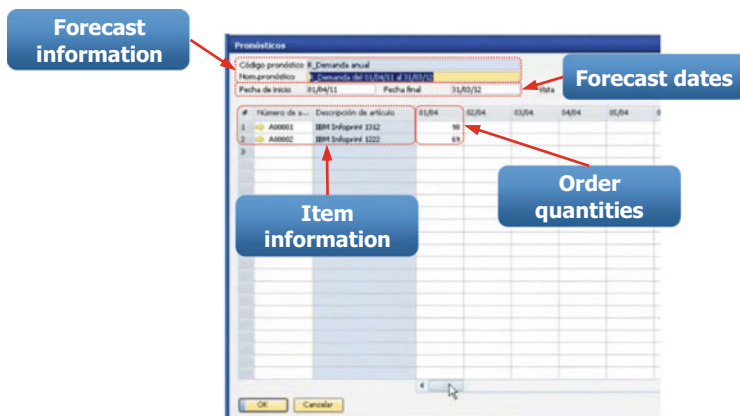


Fig. 9 Forecast generated by the MIOPPSC algorithm that will be used in the MRP module

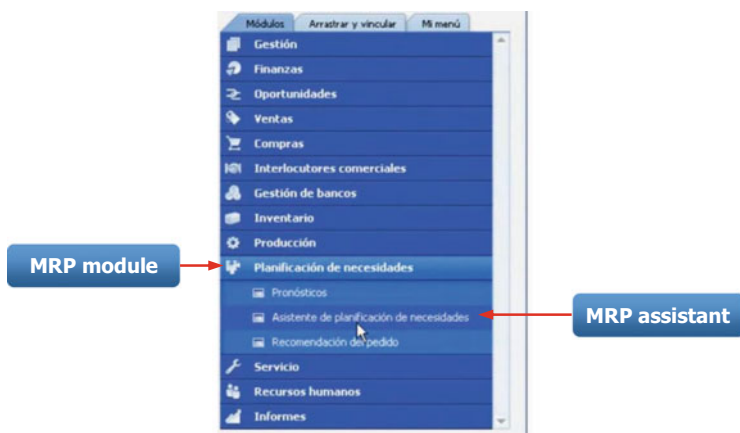


Fig. 10 MRP assistant module within SAP Business One

$x_{1,t} = \{98, 0, 97, 0, 87, 0, 34, 67, 45, 67, 79, 56\}$, with a total cost of 1,033 €.

$x_{2,t} = \{69, 29, 36, 61, 61, 26, 34, 67, 45, 67, 79, 56\}$, with a total cost of 1,234 €.

Once the solution is obtained, we now can generate a new forecast (optimized with the algorithm solution) that contains the order quantities to be replenished at the beginning of each period, that is, the first day of each month (see Fig. 9). This forecast will be used as an input by the MRP module.

The next step involves using the MRP module of SAP Business One. This module has an assistant that will guide the user through the process to obtain a procurement planning based on the previous forecast (Fig. 10).

The first step of the process consists of creating a new scenario with a name and a description (see Fig. 11). This scenario will return the procurement planning for a whole year based on the forecast computed by the MIOPPSC algorithm.

In the next step, the user has to enter all the parameters regarding the horizon planning, the range of item codes, and the visualization options (see Fig. 12):

MRP assistant

Asistente de planificación de necesidades

Seleccionar escenario nuevo o existente
Para iniciar un escenario nuevo, selec. Escenario nuevo. Para ejecutar uno existente, selec. Escenario existente. Si selecciona Escenario nuevo, escriba un nombre único.

☒ Crear escenario nuevo
☐ Seleccionar escenario existente

Nombre escenario: Pedido anual

Descripción: Pedido anual

Cancelar < Atrás Siguiente > Ignorar

Fig. 11 Creating a new scenario using the MRP assistant

Scenario details

Asistente de planificación de necesidades - Pedido anual

Detalles de escenario
Define el horizonte de planificación y los artículos que desea incluir en el informe MRP, y visualización.

Descripción: Pedido anual

Horizonte de planificación: 1

Visualizar datos en periodos: 1

Fecha de inicio: 10/03/11

Fecha final: 30/04/12

Longitud de informe: 418

Máximo fabricación acumul: 418

Tener en cuenta vacaciones para:
☒ Planificación de producción
☐ Planificación de compra

Artículos
Código: De A00001 A A00002

Grupo de artículos: Todos

Visualizar preferencias
Clasificar por: Secuencia de ensamble
☐ Visualizar artículos sin requisitos

Grabar escenario

Cancelar < Atrás Siguiente > Ignorar

Fig. 12 Scenario parameters concerning horizon planning, item codes and visualization options

- *Period dates*: the initial and final dates of the planning. These parameters are only needed for visualization purposes of the final results.
- *Item codes*: range of items to be considered by the MRP.

Finally, the last step involves providing the data source that will be used by the MRP to generate the procurement planning. As it is shown in Fig. 13, the user can select among several choices, with most of them left unselected:

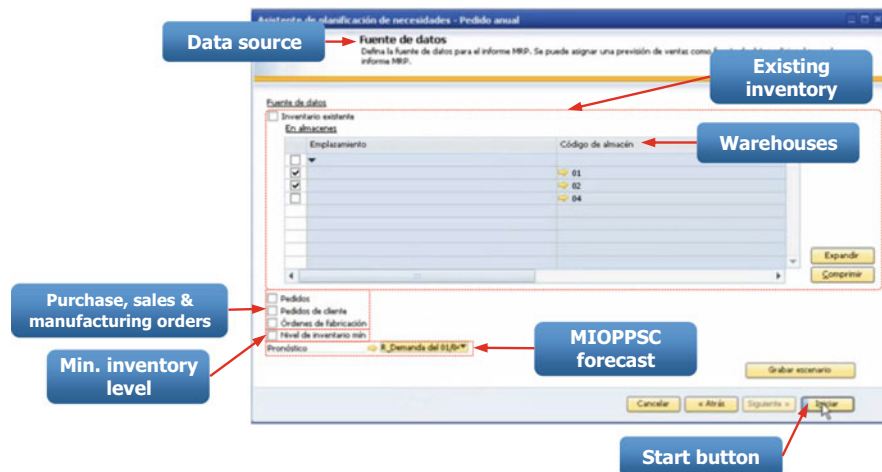


Fig. 13 Different data sources that can be used as inputs by the MRP

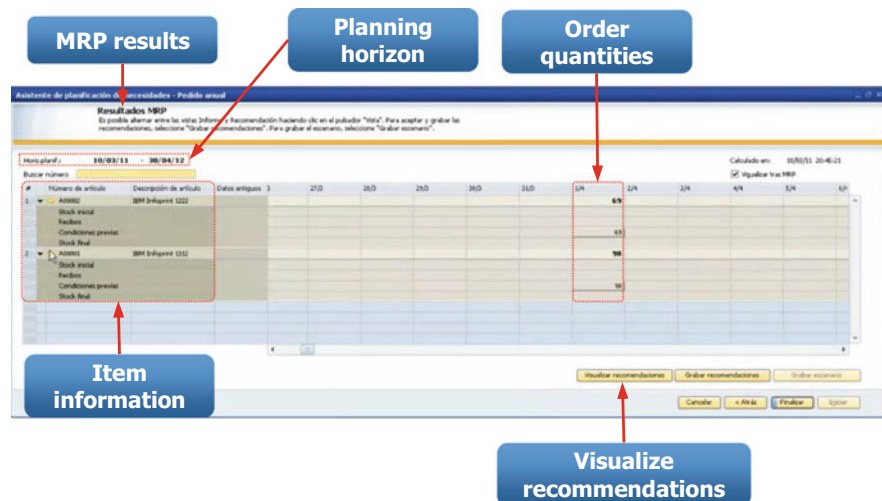


Fig. 14 MRP results using the forecast of the MIOPPSC algorithm

- *Existing inventory* (omitted): current inventory held in the different warehouses.
- *Orders of purchase, sales or manufacturing* (omitted): current orders that are already stored in the system.
- *Min. inventory level* (omitted): use the minimal inventory level as the input.
- *Forecast*: in this example, the input is sourced from the previous forecast computed by the MIOPPSC algorithm.

Once all the information needed by the MRP module has been supplied, the user can press the *Start* button to generate the MRP output.

Fig. 14 shows the MRP results for our example:

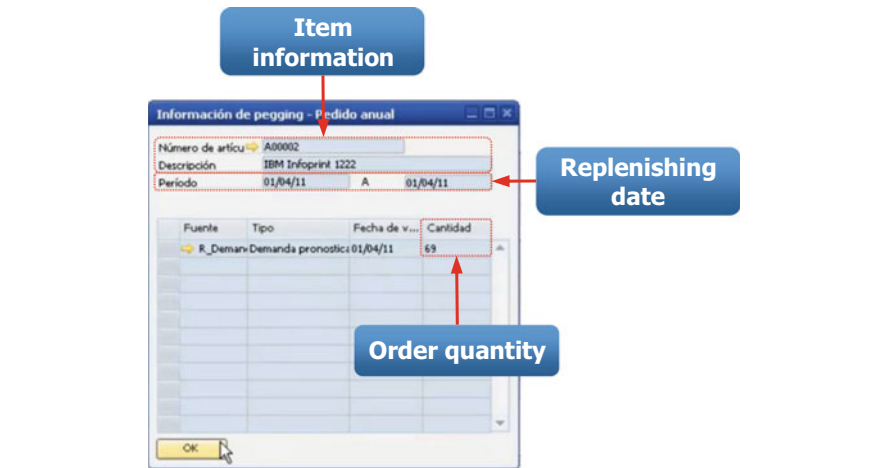


Fig. 15 Detailed view of the first item’s replenishing order

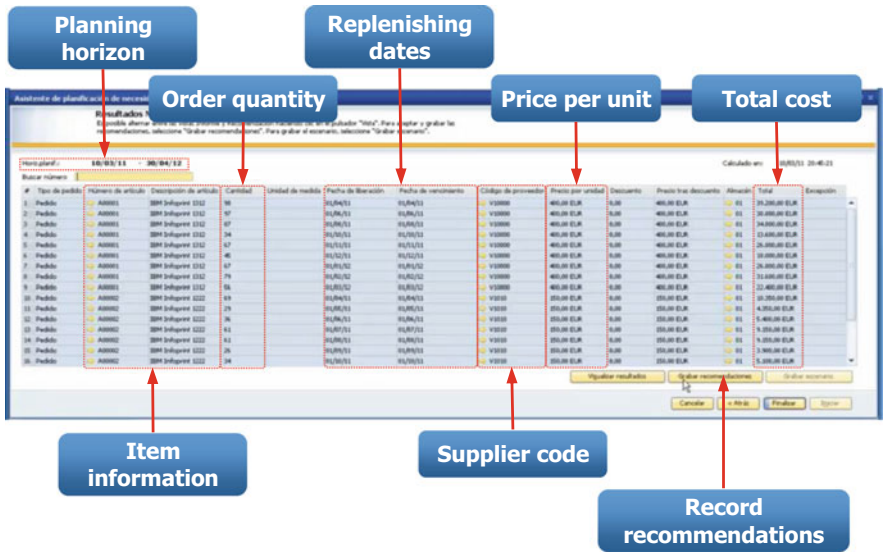


Fig. 16 Procurement recommendations generate by the MRP module

- *Planning horizon:* is the same that we set up in the scenario details.
- *Order quantities:* these orders match the ones obtained or each item by the MIOPPSC algorithm, and must be replenished at the beginning of each period (as shown in the figure).
- *Item information:* such as the item code, the description, and the stock levels.

If the user clicks on any order quantity, a new window will display showing a detailed view of the replenishing information. Fig. 15 shows the replenishing information for the first item (69 units).

In case the user presses the *Visualize recommendations* button (Fig. 14), a new window will show a full list containing all the procurement recommendations generated by the MRP module (see Fig. 16).

These recommendations can be recorded in the system to make them effective at any time by pressing the *Record recommendations* button.

6 Conclusions

We have presented a successful collaboration between a university's research group and a SAP partner company to both develop a solution in order to reduce the inventory costs in the SME, and to integrate it as an add-on in SAP Business One. As a result of this collaboration, an efficient algorithm was devised and embedded into the ERP system. Besides, the experimental results reported that the heuristic solutions were only on average a 5 % above the best solution given by a commercial solver.

The major benefits of this implementation are:

- Total integration with a high level ERP as SAP Business One.
- Provides a procurement schedule in advance to avoid stock shortages as well as overstocks.
- Optimizes the management of product costs with regards to stock holding, replenishment and delivery service to the customers.

Acknowledgments This work is partly supported by the former Spanish Ministry of Science and Innovation through the Research Project with reference MTM2010-18591 and by the new Spanish Ministry of Economy and Competitiveness through the Research Project with reference MTM2013-43396-P.

References

1. Christopher M (2011) Logistics and supply chain management. Pearson Education, Great Britain
2. Guasch JL, Kogan J (2006) Inventories and logistic costs in developing countries: levels and determinants—A red flag for competitiveness and growth. *Revista de la Competencia y de la Propiedad Intelectual*. Lima, Perú
3. Wagner HM, Whitin TM (1958) Dynamic version of the economic lot size model. *Manag Sci* 5 (1):89–96
4. Veinott AF Jr (1969) Minimum concave-cost solution of leontief substitution models of multifacility inventory systems. *Oper Res* 7:262–290
5. Zangwill WI (1966) A deterministic multiproduct multifacility production and inventory model. *Oper Res* 4:486–507
6. Love SF (1973) Bounded production and inventory models with piecewise concave costs. *Manag Sci* 20(3):313–8

7. Gutiérrez J, Sedeño-Noda A, Colebrook M, Sicilia J (2003) A new characterization for the dynamic lot size problem with bounded inventory. *Comput Oper Res* 30:383–395
8. Gutiérrez J, Sedeño-Noda A, Colebrook M, Sicilia J (2007) A polynomial algorithm for the production/ordering planning problem with limited storage. *Comput Oper Res* 34(4):934–937
9. Florian M, Lenstra JK, Rinnooy Kan AHG (1980) Deterministic production planning: algorithms and complexity. *Manag Sci* 26(7):669–679
10. Minner S (2009) A comparison of simple heuristics for multi-product dynamic demand lot-sizing with limited warehouse capacity. *Int J Prod Econ* 118:305–310
11. Dixon PS, Poh CL (1990) Heuristic procedures for multi-item inventory planning with limited storage. *IIE Trans* 22(2):112–123
12. Federgruen A, Tzur M (1991) A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time. *Manag Sci* 37(8):909–925
13. Wagelmans A, Hoesel SV, Kolen A (1992) Economic lot sizing: an $O(n \log n)$ algorithm that runs in linear time in the Wagner–Whitin case. *Oper Res* 40(1):145–156
14. Aggarwal A, Park JK (1993) Improved algorithms for economic lot size problems. *Oper Res* 41(3):549–571
15. Günther HO (1991) Bestellmengenplanung aus logistischer sicht. *Z Betriebswirtschaft* 51:541–555
16. Åxsäter S (1980) Economic lot sizes and vehicles scheduling. *Eur J Oper Res* 4:395–398
17. Gutiérrez J, Colebrook M, Abdul-Jalbar B, Sicilia J (2013) Effective replenishment policies for the multi-item dynamic lot-sizing problem with storage capacities. *Comput Oper Res* 40:2844–2851
18. Iris C, Yenisey MM (2012) Multi-item simultaneous lot sizing and storage allocation with production and warehouse capacities. In: Hu H, Shi X, Stahlbock R, Voß S (eds) ICCL'12: Third international conference on computational logistics, Shanghai, China, September 2012. Lecture notes in computer science, vol 7555. Springer, Berlin, p 129–141
19. MVS (2014) MS Visual Studio. <http://msdn.microsoft.com/en-us/vstudio/aa718325>. Accessed 1 Sep 2014
20. SAP BO SDK (2014) SAP Business One SDK General. <http://scn.sap.com/docs/DOC-28739>. Accessed 1 Sep 2014



<http://www.springer.com/978-3-319-11414-9>

Advanced Business Analytics

García Márquez, F.P.; Lev, B. (Eds.)

2015, XII, 243 p. 88 illus., Hardcover

ISBN: 978-3-319-11414-9