ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

ETSEIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

Master Thesis

OPTIMAL LOCAL INVENTORY MANAGEMENT IN A DECENTRALIZED SUPPLY CHAIN

Carried out at the Laboratory for Production Management and Processes, EPFL

by

Lidia Argilaguet Montarelo

Project Supervisor: Rémy Glardon

Project Supervisor: Nicolas Zufferey

July 2014

Contents

Acknowledgements

I would like to express my very great appreciation to Professor Glardon and Professor Zufferey, my project supervisors, for their constructive and valuable suggestions during the project, also for their patience and their enthusiastic encouragement.

Special thanks also to Christos Tsagkalidis, who together with Prof. Glardon allowed me to participate on the XBeerGame sessions to get to know the platform and helped me to understand the program.

I would also like to extend my thanks to Jaime Farrés, a student at the department carrying his master project, for his tireless support.

I would also thank the staff of the Laboratory for Production Management and Processes for hosting me during the project.

I am also grateful to friends and family who encouraged during the project.

List of Figures

List of Tables

Introduction

Motivation of the project

This project is motivated by the course Management of Enterprise Networks and Supply Chain Management taught at École Polytechnique Fédérale de Lausanne. In this course the students have to find the best/optimal strategy for their supply chain using the XBeerGame platform. This platform simulates a 4 echelon supply chain.

The best strategy is the one that minimizes their total cost and the local cost of each echelon. That is not usually the case for a real enterprise, sometimes; the echelons are not in the same enterprise. Usually when the echelons are not part of the same enterprise usually incurs in more cost because each echelon tends to optimize their cost at expenses of the other ones.

The aim of this project is to find the optimal replenishment policy that will minimize the total cost of the supply chain.

Research goals and approach

The goals are to find the best solution for two particular cases, so a discrete-event simulator will be created and a local search program to find the best solution. Hopefully this will serve as a basic path for future researchers that want to study how the variations of the data (demand, costs, and lead times) affect the total cost of the chain.

The approach will be creating a discrete-event simulator for each echelon that will represent the interactions over the time-steps of them. After validating the simulator, a local search program will be created in order to find the best solution.

Thesis outline

The thesis outline is the following; first a brief literature review will be made. Then, the problem and its characteristics will be shown. After that, the models and the methodology will be explained with its assumptions and the cost model. Then the analytical cost model will be computed for a single echelon. After that, the simulation model will be code, and then the optimisation program that will find the best solution. Those programs will be programmed using C++ language.

When the results are found a sensitivity analysis will be made to validate the parameters of the optimisation program.

After the solution is found, several experiments will be done to calculate the variation of the cost caused by the variation of the service level requirement. A comparison with other local search methods will be made. The inventory and the cost of the best solution will be analysed.

After that, the two cases analysed will be compared, and discussed.

Finally some improvements and extensions will be recommended.

Literature Review

Meta-heuristics algorithms

An overview in meta-heuristics is studied in (Blum y Roli 2003). A framework is introduced (I&D) in order to put different intensification and diversification components into relation with each other. This introduces an important thing to consider in the project, because there has to be a good relation of intensification and diversifications in the algorithm.

In (Keskin, Melouk y Meyer 2010) it is studied a vendor selection problem that integrates vendor selection and inventory replenishment of a firm. In this paper it is used the simulation-optimisation approach to solve the problem. It is build a discrete-event simulation model to evaluate the objective function. This approach will be used in this thesis. This works together with a scatter search-based meta-heuristic optimisation approach.

Several algorithms are used to find the best solution in an inventory management problem. In (Alrefaei y Diabat 2009) it is used a simulated annealing algorithm to solve a multi-objective inventory problem. In (Al-Rifai y Rossetti s.f.) it is solved the problem by decomposing the system by echelon and location, deriving expressions for the inventory policies parameters and developing an iterative heuristic optimization algorithm.

In (Sadeghi, y otros 2013) it is tackled the vendor-managed inventory using particle swarm optimization (PSO), in this proposed algorithm, a genetic algorithm (GA) with an improved operator is employed as a local searcher to turn it to a hybrid PSO. Also in (Yang, Chan y Kumar 2012) a GA is used to solve a single-warehouse multi-retailer replenishment system. In (Köchel y Thiem 2011) it is tackled the same problem (single-warehouse, multi-retailer system) by using PSO, that require using a simulation optimisation approach.

However, the previous papers do not tackle a multi-echelon supply chain system. Therefore, it was reviewed (Köchel y Nieländer 2005) where a simulation-optimisation algorithm was used to tackle a multi-echelon inventory system. Regarding the optimisation tool, it was used a genetic algorithm. Even if the problem is not exactly the same as in the project, that sets a starting point.

In (Zufferey 2012) a dynamic tabu search is studied applied to production. (Respen, Zufferey y Amaldi 2014) There are used several heuristics algorithms, greedy procedures, and local search methods, that will be useful to validate the tabu search on this project. This paper has a great contribution to the thesis on the validation chapter, since it sets the basis of the validation method procedure.

Inventory management

The literature on computing/calculating the optimal inventory policy is never ending. Publications related to the specific knowledge needed will be taken into account.

Regarding inventory management it is required to get information about inventory policies and multi-echelon management. Information about decentralized and centralized supply chain is needed as well.

In (Clark y Scarf 1960) it is tackled the multi-echelon problem analytically. This is one of the first papers that were written to face a multi-echelon supply chain. This paper uses some assumptions that will be taken in the project. For example, that the demand originates in the system at the lowest installation (retailer in the project's case) and that each echelon backlogs excess demand. It is found that the (s,S) policy is optimal for the linear-plus-fixed-ordering cost case (or a fixed ordering/setup cost).

In (Lee y Whang 1999) a decentralized multi-echelon supply chain is tackled using incentives and information. This paper discusses different performance measurement involving transfer pricing, consignment, shortage reimbursement… An infinite horizon is considered. There are two echelons in the problem and the first is facing stochastic demand. Since in the project no such measures will be taken, this paper only provides a good point of view regarding the equations of backlog cost.

The aim of that problem is delivering the desired end customer service level at minimum network inventory, with the inventory divided among the various echelons. The unmet demand at the first echelon is back ordered.

A multi-echelon supply chain with decentralized control is tackled in (Axsäter, A framework for decentralized multi-echelon inventory contorl 2000). Where there is a central warehouse and a certain number of retailers. The cost structure consists in local costs and penalty cost for a delay at the warehouse. This paper has certain relevance on the project since there are used penalty costs to try to reduce the delays at the warehouse.

In (Li 2010) it is studied the inventory behaviour of autonomous and self-serving firms in a decentralized retailer-manufacturer serial supply chain. The optimal inventory is characterized analytically with and without information sharing. Some extensions are discussed in this paper:

N-stage serial systems, batch ordering policies, fixed setup costs and Markovian customer demand.

Optimal replenishment policies for all members of the supply chain are determined by hybrid meta-heuristic algorithm recently developed (Duan, Liao y Yi 2013). The hybrid meta-heuristic generates a trial solution and supplies this candidate solution to the supply chain simulation model to evaluate its objective function value. The meta-heuristic optimizer then generates a new input set according to its intelligent searching mechanism. This paper has great importance regarding this project. The same procedure will be used in the thesis.

The (s,S) inventory policy is studied in (Baron, Berman y Perry 2010), they use the (s,S) policy for perishable items ordered in batches. For the (s,Q) policy there are two papers reviewed: (Rosling 2002) and (Axsäter, A simple procedure for determining order quantities under a fill rate constraint and normally distributed lead-time demand 2006).

1.Description of the problem

The problem consists in a decentralized supply chain. A decentralized supply chain is a chain where all the echelons work autonomously and the decision makers are distributed in each echelon. The opposite case would be a centralized supply chain, where all the decisions are taken in the headquarters.

The supply chain consists in four echelons (in downstream sense): factory, wholesaler, distributor and retailer. The product flows downstream and the information upstream. The only product commercialized is beer. The retailer is the echelon that faces the stochastic demand from the market.

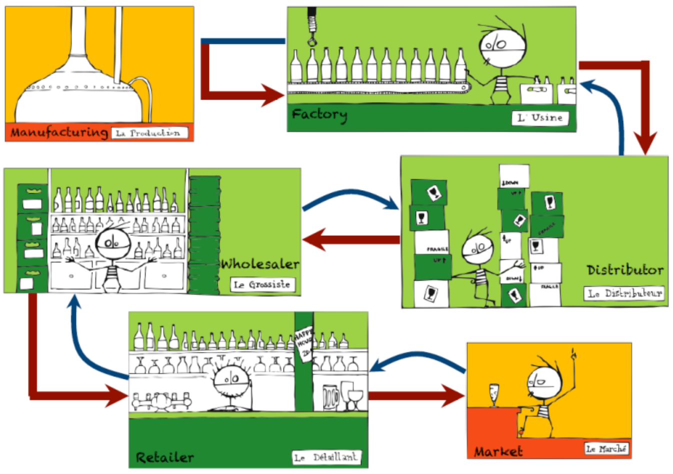


Figure 1.1: Illustration of the beer distribution in the XBeerGame[1]

The objective is to minimize the cost of the total chain (global optimisation). The problem is complex and there are interactions between the echelons each unit of time. Therefore, a simulation-driven optimisation is chosen. Also, at it can be seen in the literature review, this is the most usual approach for problems with more than one echelons in the chain.

In principle, the XBeerGame platform is supposed to be used, but as its performance does not match the requirements, it will not be used. It is proposed to code a new simulation program in order to use it to find the total cost of the chain. That means that the simulation program will compute the objective function value that will be used in the optimisation program.

[1] XBeerGame, Université Laval

2.Models and Methodology

Chain

As said in the previous chapter, there are four echelons and two main flows in the system.
2.1There are the information flows and the product flows.

In the Figure 2.1 is shown the Information and product flow of the chain. The red arrows represent the orders received and sent of each echelon in each time step (O_t^i). That would be the information flow.

The green arrows represent the product shipment from each echelon to the downstream echelon in each time step (S_t^i).

At each echelon it is represented as well the stock information that each echelon would know in a real case. The L_i with $i = M, F, D, W, R$ is the lead time of each echelon (Manufacturer, Factory, Distributor, Wholesaler and Retailer). In this problem the manufacturer is integrated in the factory.

The on-hand inventory[2] is represented by ax_t^i and x_t^i, where the first is the initial on-hand stock at time step t and the latter is the on-hand stock at the end of the time step t.

The backlogged orders[3] at echelon i at time step t are represented by bx_t^i.

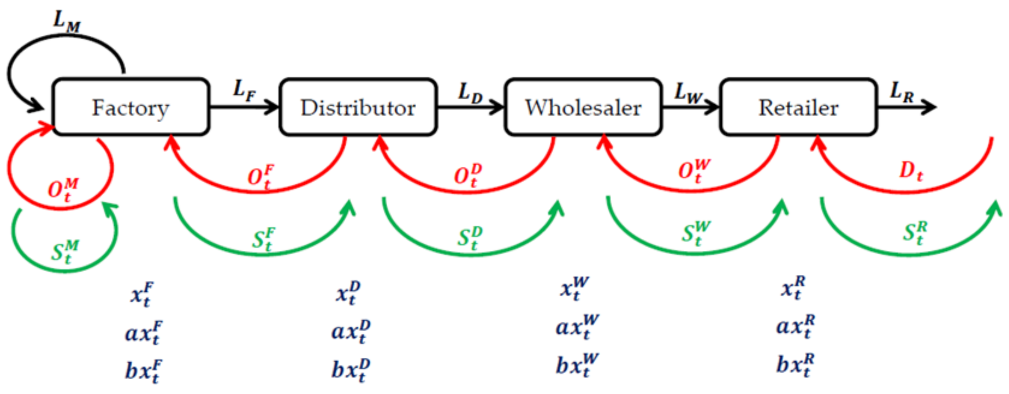


Figure 2.1: Scheme of the supply chain

[2] On-hand inventory: Physical inventory (stock) at the echelon
[3] Backlogged orders: Orders that have not been satisfied in this time step or in the previous ones

2.2 Inventory policies

It has been explained the problem and the chain. But now is the time to explain the methodology that the chain will use. As each echelon has to have a policy to manage the orders and the shipments an inventory policy has to be used.

2.2.1 Event succession

An event succession has to be defined; the same will be for the two inventory policies. And the same succession will be used in all the echelons.

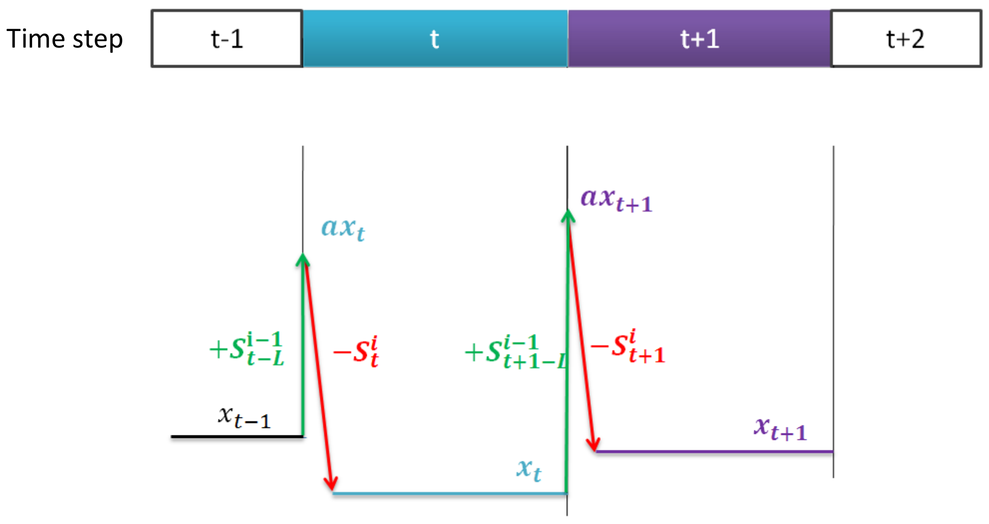


Figure 2.2: Scheme of the event succession

The succession of the events is the following:

1. Echelon i receives the shipment from the previous echelon
    a. On hand inventory $(ax_t) = x_{t-1} + S^{i-1}_{t-L}$
2. Echelon i sends a shipment to the following echelon
    a. Final inventory (x_t)= $ax_t - S^{i}_{t}$

These steps happen when there is enough quantity to deliver on time. In that case, the shipment received in every echelon is the shipment due at time t, it is shown in the sub index of S^{i-1}_{t-L}. It can be that the shipment is bigger than the order received $(S^{i-1}_{t-L} > O^{i-1}_{t})$, this means that at this period backlogged quantity has been supplied.

In the case that there is not enough quantity to deliver the order, that quantity is backlogged $(bx = bx + O_t)$. The backlogged quantity is supplied as soon as there are enough pieces.

To manage the orders placement, two different inventory policies will be studied.

2.2.2 Reorder point method

The reorder point method it also named (s, Q) in this project. According to this policy, there are two parameters to make a decision regarding the orders.

In this method the orders are placed when the inventory position[4] drops below s. So s is the reorder point. As in this method the order quantity is fixed, the echelons will order a quantity Q. On the graph below (Figure 2.3), the lead time is 3 days, the daily demand is a normal distribution with a mean of 100 pieces and a std. dev. of 5, the reorder quantity is 400.

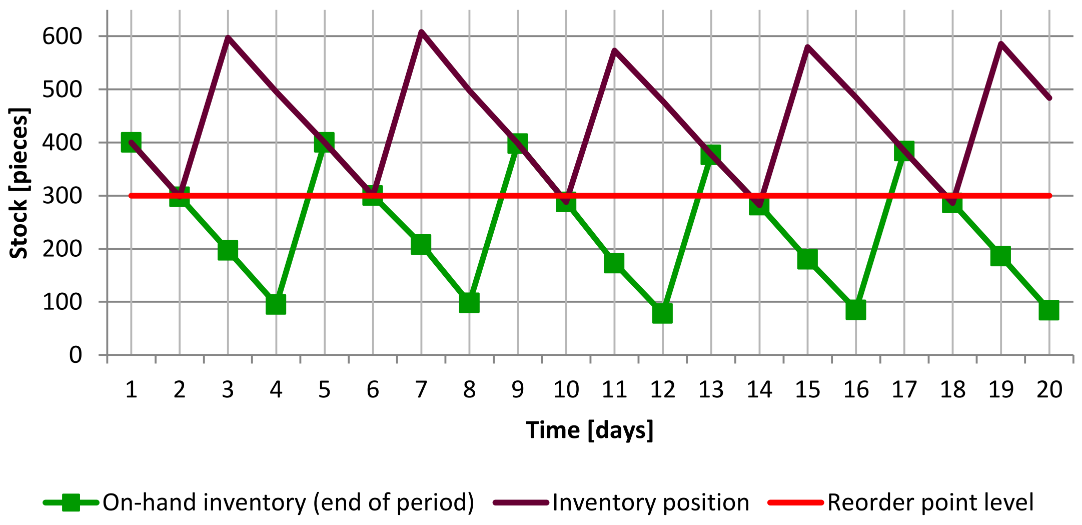


Figure 2.3: Reorder point method (s,Q)

2.2.3 (s,S) policy

This is the other policy that will be studied in this project, also named for now on as (s, S) policy. As the previous policy there are two decision parameters. Orders are placed whenever the inventory position drops below s (as the (s, Q) policy). But the order quantity is variable, as it is ordered the difference between the inventory position and the upper level (S). The demand is the same as in the (s,Q) case, the low level s is 300, and the upper level S is 600 pieces. It can be observed that that leads to more variation on the on-hand stock.

[4] Inventory position: on-hand stock + standing orders – backlogged orders

As it can be seen in the following figure (

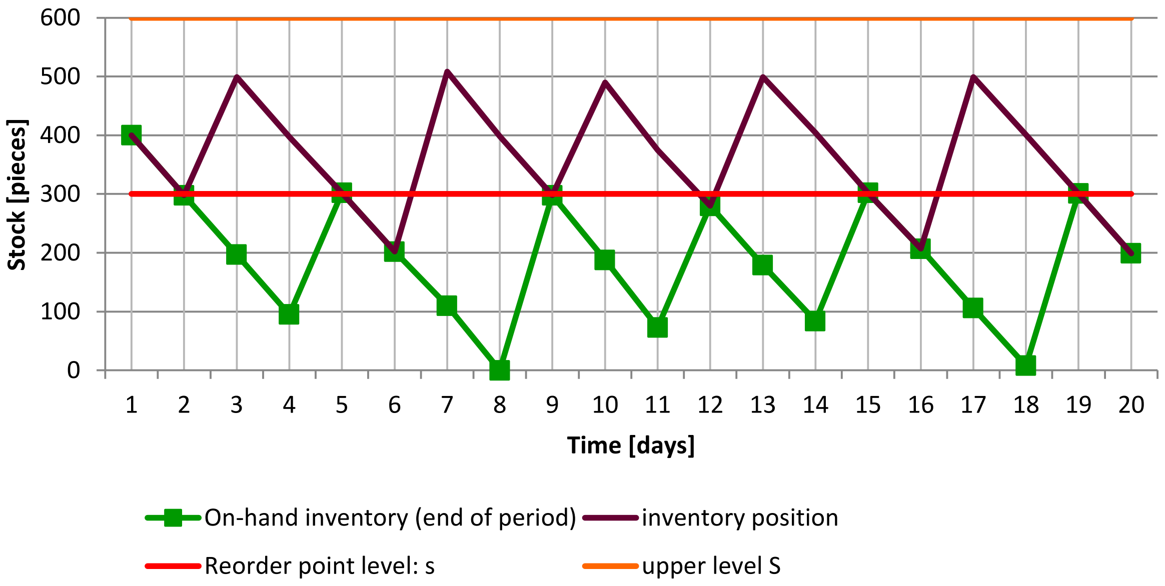


Figure 2.4) at time 1 and 2 it is ordered the difference between S and Inventory position.

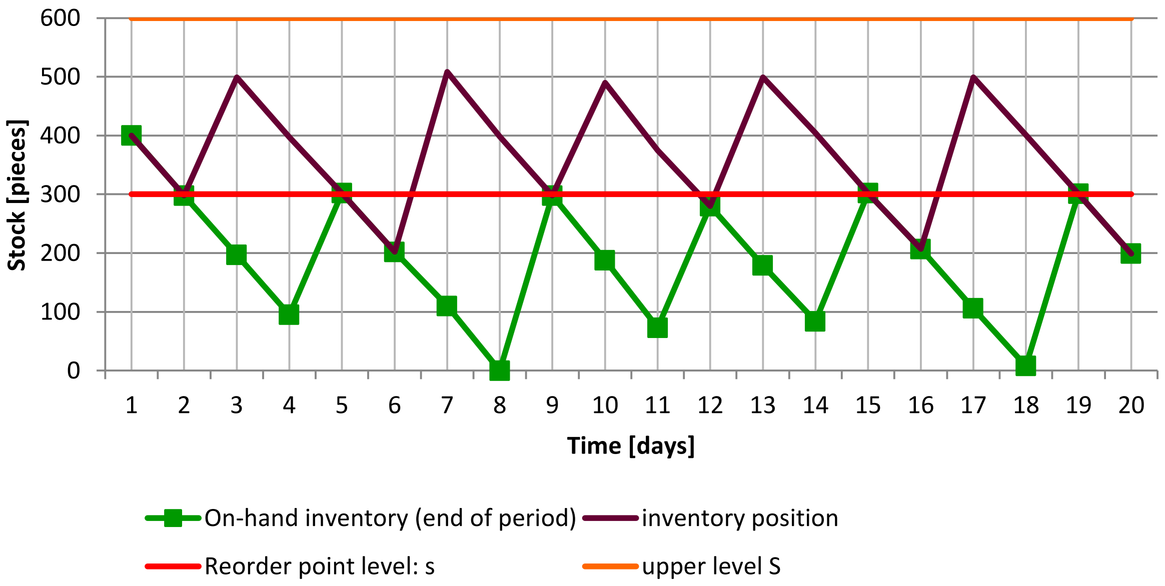


Figure 2.4: (s,S) policy

2.2.4 Assumptions

In order to be able to simulate the processes some assumptions were made. The assumptions were either to make the simulation similar to the one in the XBeerGame or to compute the cost in a straight-forward way.

Regarding the inventory management:

- There are not partial deliveries, nor missed orders.
- The same type of replenishment system is used, but, obviously, with different parameters.
- The lead time of the pieces manufacturing is constant, and the factory has unlimited capacity.
    - That means that it delivers on-time always all the orders placed by the factory.
- The lead time (transportation time) is constant for all echelons.
    - That does not mean that all the pieces are supplied when they are ordered.

2.3 Cost model

The aim of the project is to find a solution with the minimum total cost. In order to do that, the cost model has to be defined.

As the total cost will be computed, the money trespassed to the echelons from other echelons has a balance of zero. These are the costs of buying pieces, the cost that comes from the price of the parts.

The fixed cost will not be taken into account, these are the costs that are independent of the volume sold or bought.

The manufacturing cost is not taken into account as it would be similar in all the cases as it relates directly to the pieces that the market demands.

The costs taken into account are:

- Launching cost ($\mathrm{LCost_i}$): cost of placing an order
- Backlog cost ($\mathrm{BCost_i}$): cost of having an order backlogged
- Holding cost ($\mathrm{HCost_i}$): cost of having pieces in the storage (inventory)

The holding cost will be computed using the stock on-hand at the end of the period, for the back orders as well. The backorder cost will be computed with the backorders accumulated at the end of the period.

In the following table (Table 2.1) the unit costs are shown. The costs were used in the calibration of the XBeerGame during the course.

Table 2.1: Costs of the different echelons

	Factory	**Distributor**	**Wholesaler**	**Retailer**

Ordering cost	$[\mathrm{m.u.}]$	25	25	25	25
Inventory cost	$\left[\frac{\mathrm{m.u.}}{\mathrm{pieces}\cdot\mathrm{day}}\right]$	0.2	0.4	0.6	0.8
Back order cost	$\left[\frac{\mathrm{m.u.}}{\mathrm{pieces}\cdot\mathrm{day}}\right]$	0.2	0.3	0.6	1

So the total cost is computed in the following form (Eq. 2.1).

$$Total\ cost = \sum_{i=1}^{4} LCost_i + \sum_{i=1}^{4} BCost_i + \sum_{i=1}^{4} HCost_i \qquad \text{(Eq. 2.1)}$$

The demand is constant and it follows a normal distribution with a mean of 3000pcs. and a standard deviation (σ_s) of 150pcs. To the retailer arrive 3 different orders during one day.

3.Analytical model of the cost

According to the literature the optimum value for a single echelon for the (s,Q) policy can be found. That is assuming there are not backlog costs. This is interesting in the sense that a good initial solution can be computed to initialize the optimisation problem. This is the so called Economic Order Quantity method. That computes the optimum quantity Q balancing que holding cost and launching costs.

The total cost (Eq. 3.1) for one echelon for the entire horizon is:

$$C_{total} = Np + \frac{Q}{2}h + \frac{N}{Q}L \qquad \text{(Eq. 3.1)}$$

Where:

N: Total demand per period $[\frac{\text{pieces}}{\text{t.s.u}^{5}.}]$

p: Unit price $[\text{m.u.}^{6}]$

Q: Order quantity $[\text{pieces}]$

h: Holding cost $\left[\frac{\text{m.u.}}{\text{t.s.u.}\cdot\text{pieces}}\right]$

L: Launching cost $[\text{m.u.}]$

Deriving the (Eq. 3.1) to find the minimum, the optimum Q is found (Eq. 3.2):

$$Q = \sqrt{\frac{2NL}{h}} \qquad \text{(Eq. 3.2)}$$

According to the Reorder Point Method, the reorder point s can be found using the equation (Eq. 3.3):

$$s = SafetyStock + L \cdot D \qquad \text{(Eq. 3.3)}$$

Where:

L: Lead time $[\text{t. s. u.}]$

D: Demand in one time step $\left[\frac{\text{pieces}}{\text{t.s.u.}}\right]$

In the following table (Table 3.1) the optimum quantity is computed using (Eq. 3.2) and (Eq. 3.3). It is computed assuming their demand is constant and also is constant the lead time of the orders. The reorder point is computed as well but without taking into account the safety stock.

[5] t.s.u.: Time step units
[6] m.u.: Monetary units

The lead time of every echelon is the lead time that they have to receive the orders. That takes into account the transportation time and the processing delay of the order (1 day). For example the lead time for the wholesaler is 4 days (3 days transportation from the distributor + 1 day of delay).

Table 3.1: Optimum quantities and reorder points for the echelons

		Factory	Distributor	Wholesaler	Retailer
Lead time	[days]	4	5	4	3
Q	[pieces]	866	612	500	433
s	[pieces]	12000	15000	12000	9000

4.Simulation model (s,Q) and (s,S)

4.1 Description

A simulation model is coded to compute the cost of all the echelons and to simulate their interactions. The simulation will follow the steps of the events as explained in the chapter 2.2.1 Event succession. In this chapter the material flow is shown. The information flow (I) and material flow (M) together follow this process:

1. Shipment received (M)
2. On-hand inventory updated (I)&(M)
3. Order received (I)
4. Shipment sent (M)
5. On-hand inventory at the end of the period is computed (I)&(M)
6. Orders are placed (I)

It is important to notice that there is a main function that simulates one period of time with this process. This function is repeated 4 times to simulate the 4 echelons and then t_{end} times to simulate for echelons over a period of t_{End}.

The shipments are sent with backorders as a priority, and if they are fully satisfied, the newly received orders. So the orders are accumulated and are satisfied following a FIFO[7] queue.

As it is needed to compute the service level of the retailer, the quantity sent on-time is computed at each time step.

The difference between the code of (s,S) and (s,Q) is only when the orders have to be placed. The initial inventory for both methods is the same and it is shown on Table 4.1. This initial inventory used is one of a calibration of the XBeerGame used during the course. As it goes downstream, the value is higher due to the stochasticity that may face the retailer. It is important to say as well, that this value can be changed, as it does not affect in a long term cost, as the on-hand value gets stable around the same value no matter the value of the initial inventory.

[7] FIFO: First in, first out.

Table 4.1: Initial stock at each echelon

		Factory	Distributor	Wholesaler	Retailer
Initial inventory	$[\text{pieces}]$	10000	12000	14000	16000

To simulate the demand, it is divided in 3 orders a day, the total demand is D_T (Eq. 4.1) and the three orders are D_1, D_2 and D_3. The demand D_1 is the minimum of D_T and a normal distribution of $\mathrm{N}(1000{,}100)$ (Eq. 4.2), that is just in case that $\mathrm{N}(1000{,}100)$ is greater than the D_T. So the demand D_1 will be D_T and the others 0 (Eq. 4.3) and (Eq. 4.4). That is to make sure that $D_1 + D_2 + D_3 = D_T$.

$$D_T = \mathrm{N}(3000{,}150) \quad \text{(Eq. 4.1)}$$

$$D_1 = \mathrm{MIN}\{\, \mathrm{D_T},\ \mathrm{N}(1000{,}100)\,\} \quad \text{(Eq. 4.2)}$$

$$D_2 = \mathrm{MIN}\{\, D_T - D_1, \mathrm{N}(1000{,}100)\,\} \quad \text{(Eq. 4.3)}$$

$$D_3 = \mathrm{MAX}\{\, D_T - D_2 - D_1,\ 0\,\} \quad \text{(Eq. 4.4)}$$

4.2 Pseudocode

The notation followed to write the pseudocode is at Table 4.2.

Table 4.2: Notation of the pseudocode

t	Time step $\boldsymbol{t}$
endt	Duration of the simulation
$\mathrm{O}_\mathrm{t}^\mathrm{i}$:	Orders placed by echelon $\boldsymbol{i-1}$ to echelon $\boldsymbol{i}$ at time $\boldsymbol{t}$
$\mathrm{S}_\mathrm{t}^\mathrm{i}$:	Shipment sent from echelon $\boldsymbol{i}$ to echelon $\boldsymbol{i-1}$ at time $\boldsymbol{t}$
L_i:	Transport lead time from echelon $\boldsymbol{i}$ to echelon $\boldsymbol{i-1}$
$\mathrm{x}_\mathrm{t}^\mathrm{i}$:	Inventory level at echelon $\boldsymbol{i}$ at the end of the period $\boldsymbol{t}$
$\mathrm{ax}_\mathrm{t}^\mathrm{i}$:	Available inv. at echelon $\boldsymbol{i}$ at the beginning of the period $\boldsymbol{t}$
bx^i:	Backlog vector of echelon $\boldsymbol{i}$ following a FIFO queue
$\mathrm{Q}_{\mathrm{on\ time}}$	Cumulated quantity delivered on time from the retailer (echelon $\boldsymbol{i=1}$) to the market $(\boldsymbol{i=0})$
Sup_i	Replenishment upper level of $(\boldsymbol{s},\boldsymbol{S})$ policy of echelon $\boldsymbol{i}$
s_i	Reorder point of $(\boldsymbol{s},\boldsymbol{S})$ and $(\boldsymbol{s},\boldsymbol{Q})$ policy of echelon $\boldsymbol{i}$
Q_i	Order quantity of $(\boldsymbol{s},\boldsymbol{Q})$ policy of echelon $\boldsymbol{i}$
h_i	Inventory holding cost at echelon $\boldsymbol{i}$ $\left[\frac{m.u}{piece\cdot t.s.u.}\right]$
$\mathrm{HCost}_\mathrm{i}$	Cumulated inventory holding cost at echelon $\boldsymbol{i}$ $[m.u.]$
b_i	Backorders cost at echelon $\boldsymbol{i}$ $\left[\frac{m.u.}{piece\cdot t.s.u.}\right]$
$\mathrm{BCost}_\mathrm{i}$	Cumulated backorder cost at echelon $\boldsymbol{i}$ $[m.u.]$

l_i	Launching cost at echelon $\boldsymbol{i}$ $\left[\frac{m.u.}{piece}\right]$
$\mathrm{LCost_i}$	Cumulated launching cost at echelon $\boldsymbol{i}$ $[m.u.]$
$\mathrm{Suma(x)}$	This function sums up all the elements of the vector x

```
Initialization (t=0)
$i = r, w, d, f$
The initial orders are 0 for all echelons
The initial shipments are 0 for all echelons
The initial backorders are 0 for all echelons
The initial inventory at the echelons is set as the values in Table 4.1.
While $(\mathbf{t} \leq \mathbf{endt})$, do:
  For each echelon do:
      The on-hand inventory is updated with the shipments due today
    • Set $ax_i[t] = x_i[t-1] + S_{i-1}[t - L_{i-1}]$
    ➢ If the orders and all the backorders can be fulfilled do:
        o Set $S_i[t] = O_i[t] + suma(bx_i)$
        o Set $x_i[t] = ax_i[t] - (O_i[t] + suma(bx_i))$
        o Clear vector $bx_i$
        o Set $Q_{on\ time} = Q_{on\ time} + O_i[t]$
    ➢ Else, if they cannot be fully fulfilled
        o Satisfy the backorders in a FIFO order
        o Compute $S_i[t], x_i[t], bx_i$ and $Q_{on\ time}$
     The new orders are placed
    • Set $InvPos_i :=$ on − hand inventory + standing orders − back orders
    ➢ If $(InvPos_i < s_i)$
        ○ Set $O_{i-1}[t] = Q_i$ (case (s,Q))
        ○ Set $O_{i-1}[t] = Sup_i - InvPos_i$ (case (s,S))
        ○ Set $Lcost_i = Lcost_i + l_i$
    ➢ Else
        ○ Set $O_{i-1}[t] = 0$
    The cost at the end of the period is computed
    • Set $HCost_i = HCost_i + h_i(x_i[t])$
    • Set $BCost_i = BCost_i + b_i \cdot suma(bx_i)$
```

- Set $S_0[t] = O_0[t]$[8]
- $t = t + 1$

Final updates:

- Set $\text{ServiceLevel} = \frac{Q_{\text{on time}}}{\text{Accumulated demand at retailer's}}$
- Set $\text{Total cost} = \sum \text{BCost}_i + \sum \text{LCost}_i + \sum \text{HCost}_i$

4.3 Sensitivity analysis: simulation duration

A sensitivity analysis over the simulation duration is performed. As is important to have reliable results with one run of the simulation, 10 runs are performed for different cases. So the simulation was launched 10 times over a certain periods of time ($t = 100, 200, 300, 500, 750, 1000\ days$). In order to decide when to stop the simulation, two graphs are made: with the service level over time and the cost over time. The cost is divided over the period of time, so they can be compared among them.

For one case ($[s, Q] = \{10592, 11992, 15200, 11976, 361, 539, 615, 916\}$) 6 different end times and 10 runs each one were made. The values of the case represent the 4 reorder points of the 4 echelons (retailer to factory order) and the 4 order quantities. It can be notice that as longer the simulation is the less variance there is (Figure 4.1 and Figure 4.2).

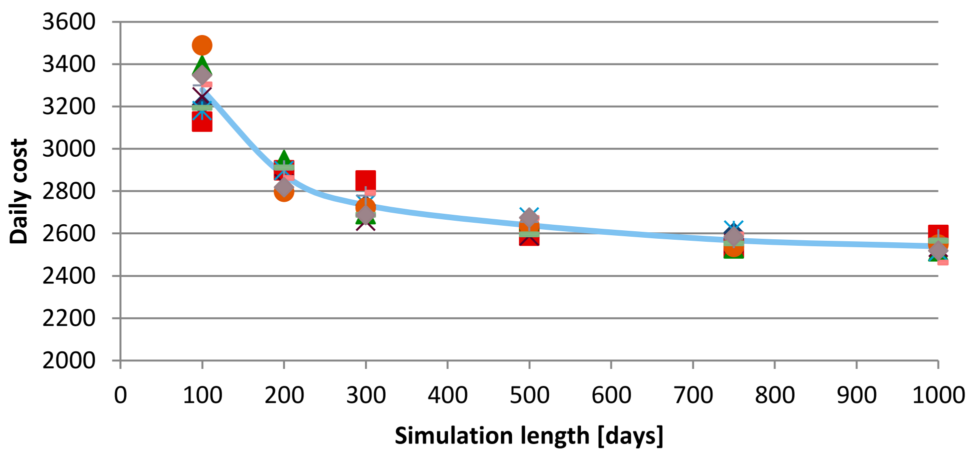


Figure 4.1: Graph of the daily cost over different simulation lengths

[8] The index 0 represents the manufacturer, who ships the material when it is ordered.

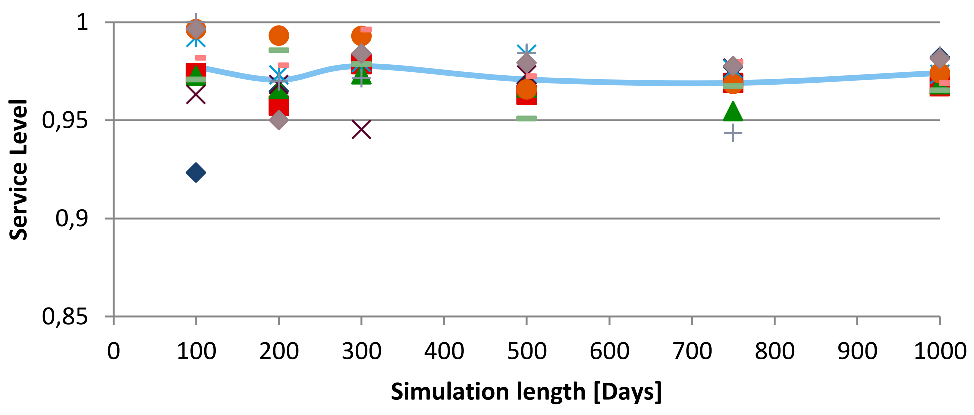


Figure 4.2: Graph of the service level over different simulation lengths

A table with the main statistics is made for this case to these observations can be corroborated with data. The confidence interval for the mean ($C.I. of\ 95\%$) is computed with the distribution t-student, as the real standard deviation behind the data is unknown. To see how big was the confidence interval (one half); it is divided over the mean. So it can be seen that as longer the period, the smaller the confidence interval gets with respect of the mean. With exception of $t = 1000\ days$ that is a bit greater than the previous limit times. It can happen because of the case and because it is stochastic. More cases were tested (Appendix A: Output data) to make sure that more or less with a $t = 500\ days$ the results are reliable.

Table 4.3: Statistical information of the runs

$t =$	100	200	300	500	750	1000
Mean	3278,067	2875,765	2733,408	2638,61	2567,459	2539,645
Std. Dev.	110,1069	47,06992	58,88274	35,70989	30,98973	38,90396
C.I.	78,76574	33,67179	42,12218	25,54532	22,16871	27,83022
C.I./mean	2,40%	1,17%	1,54%	0,97%	0,86%	1,10%

4.4 Computation time

It is important to know the computation time for different simulation lengths. As this is an important factor for the Tabu Search. In the following graph (Figure 4.3) certain set of s_i and S_i are used, the set of solutions used the does not have a great impact on computation time. As it can be seen, the computation time has a linear correlation with the simulation time. For example, the computation time of 1000 days divided in 10 parts is 0.54s, and for 500 days is 0.27s.

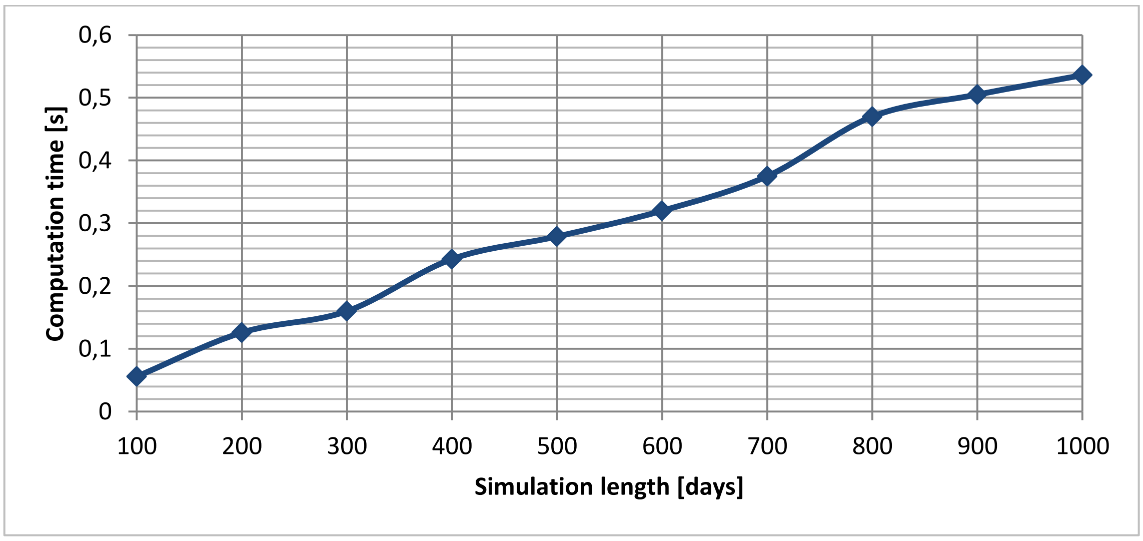


Figure 4.3: Graph of the computation time depending on the simulation length

5.Optimisation model (s,Q) and (s,S)

5.1 Description

The optimisation problem, to be resolved, various steps will be followed (Figure 5.1). First, it is needed to define the Objective Function and its restrictions. Then, input and the initial solution have to be calculated to be able to run the optimisation. The optimisation will be a simulation driven optimisation, that means that the objective function will be computed using the simulation. Finally, the output will be the decision variables.

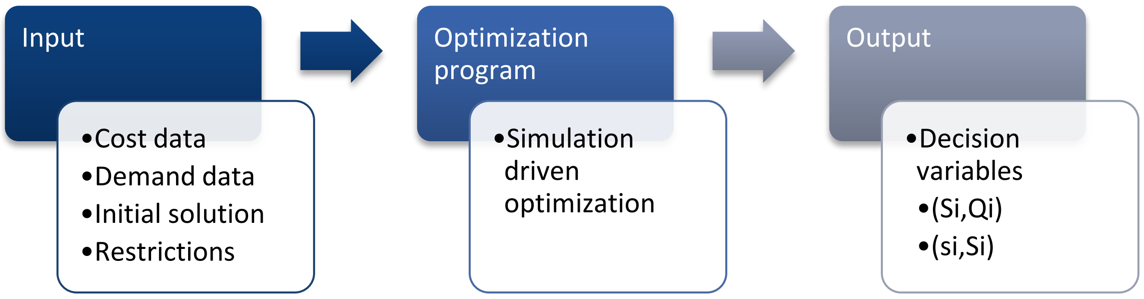


Figure 5.1: Optimisation problem steps

5.2 Optimisation problem: objective function and restrictions

The objective is to minimize the total cost of the chain. Every cost of each echelon is weighted the same, so no priorities are set. The restrictions include the service level required and the non-negativity of the variables.

As shown in chapter 2.3 Cost model, the total cost is the following:

$$Total\ cost = \sum_{i=1}^{4} LCost_i + \sum_{i=1}^{4} BCost_i + \sum_{i=1}^{4} HCost_i \qquad \text{(Eq. 5.1)}$$

Then the objective function for the (s,Q) model and the restrictions are:

$$[MIN] f(s_1, s_2, s_3, s_4, Q_1, Q_2, Q_3, Q_4) = \sum_{i=1}^{4} LCost_i + \sum_{i=1}^{4} BCost_i + \sum_{i=1}^{4} HCost_i \qquad \text{(Eq. 5.2)}$$

$$Service\ Level \geq 95\% \qquad \text{(Eq. 5.3)}$$

$$s_i > 0\ \forall i \qquad \text{(Eq. 5.4)}$$

$$Q_i > 0\ \forall i \qquad \text{(Eq. 5.5)}$$

The objective function for the (s,S) model and the restrictions are the same as the (s,Q) model with exception of (Eq. 5.8). This restriction is added to avoid ordering negative quantities since the order quantity is $S_i - s_i$.

$$[MIN] f(s_1, s_2, s_3, s_4, S_1, S_2, S_3, S_4) = \sum_{i=1}^{4} LCost_i + \sum_{i=1}^{4} BCost_i + \sum_{i=1}^{4} HCost_i \qquad \text{(Eq. 5.6)}$$

$$Service\ Level \geq 95\% \qquad \text{(Eq. 5.7)}$$

$$s_i < S_i\ \forall i \qquad \text{(Eq. 5.8)}$$

$$s_i > 0\ \forall i \qquad \text{(Eq. 5.9)}$$

$$S_i > 0\ \forall i \qquad \text{(Eq. 5.10)}$$

As said before the decision variables are s_i and Q_i for the (s,Q) model. For the (s,S) model s_i and S_i are the decision variables. With this variables the system can compute when to order and when not to.

5.3 Optimisation program

The general procedure of the tabu search will be explained in this section, and the pseudocode will take part in another chapter. The main idea of the tabu search is that it is a local search, but with forbidden moves. That means that during a certain amount of iterations it cannot go back to already evaluated solutions. This way the candidate solution does not get stuck in a local minimum. Tabu search combines intensification and diversification, regarding the intensification, the tabu search looks for the best neighbour of the best neighbour, until a local minimum. The diversification is accomplished forbidding some movements to avoid getting stuck in a local minimum.

5.3.1 Neighbourhood

The tabu search starts searching the neighbours of the initial solution. The variables are found in a vector: $[s_1, s_2, s_3, s_4, Q_1, Q_2, Q_3, Q_4]$ or $[s_1, s_2, s_3, s_4, S_1, S_2, S_3, S_4]$ for the (s,S) case. The maximum neighbour quantity that can be found is 16. The first 8 neighbours come from increasing each decision variable by a certain amount. The others 8 neighbours come from decreasing the initial solution by a certain amount. The same amount is used for increasing and decreasing the value of the variables. The increasing amount and the decreasing amount are: Δs and ΔQ or ΔS in case of the (s,S) case. If a move is forbidden, the neighbour that comes from it, it is not calculated, hence it does not get included in the neighbourhood.

5.3.2 Selection of the best neighbours

The value of the objective function and the service level of the retailer are computed for every neighbour in the neighbourhood. The neighbours with a service level smaller than 95% are deleted of the possible solutions. Then, the half better neighbours are selected.

In order to decide which one is the best, the solution of every neighbour is run 10 times. This makes the tabu search more robust.

If the best of the neighbourhood is better than the best solution found, the latter is substituted by the best neighbour.

After that, on the basis of the best neighbour, the program starts again looking for the neighbours and etc.

5.3.3 Dynamic search

The tabu search looks for the solution adding or dropping a certain value to the neighbours as explained before. As this might miss better neighbours that can be reached by a smaller value of these parameters, a dynamic tabu search is used. The dynamic tabu search work as follows: when after a certain number of iterations any better solution is found, the parameters (Δs and ΔQ or ΔS) are reduced by a factor X. If a better solution is found, the parameters Δs and ΔQ or ΔS get reset to the original values.

5.4 Initial solution

An initial solution is a very important part of the optimisation problem, because as better it is, the faster the program will found good solutions. As shown before, the optimum values for a single echelon can be found, so those values can be used to start the program.

The program saves the initial solution as the better solution ever found and then stars looking for its neighbours. In order to save a reliable value of the initial solution, it is run 10 times and then the value of the Objective function is stored. The value of 10 runs is chosen because as seen in the validation of the simulation length, within 10 runs, reliable results are found, without a great variance and a small confidence interval.

5.4.1 [s,Q] model initial solution

For this model, the initial solution is as mentioned before; in the following table (Table 5.1) the quantities Q are shown.

Table 5.1: Initial order quantity Q

		Factory	Distributor	Wholesaler	Retailer
Initial Q_i	[pieces]	866	612	500	433

For the reorder point the values depend on a safety stock. So as it is really difficult to set a safety stock close to the optimum, it is set bigger. The values of the s on the Table 5.2 do not take into account the safety stock.

Table 5.2: s values without including the safety stock

	Factory	Distributor	Wholesaler	Retailer

s	[pieces]	12000	15000	12000	9000

There is another thing important of choosing the s value: if the initial solution does not have any neighbour with a service level greater or equal than 95%. The program cannot start iterating because any of these are feasible solutions. This is why is important to set a greater value of the safety stock. After many trials the following values were chosen (Table 5.3). That is setting 1000 pieces as a safety stock. As the reorder point method takes into account a constant lead time. This safety stock accounts for the variations in the lead time and variations of the demand.

Table 5.3: Initial reordering points s_i

		Factory	Distributor	Wholesaler	Retailer
Initial s_i	[pieces]	13000	16000	13000	10000

5.4.2 [s,S] model initial solution

This model is a bit more complicated in terms of defining a feasible initial solution. The path taken was simulating the initial solution of the model [s,Q] and store the value of the inventory position at each time step. So after calculating the mean and rounded it to the thousands, the result is shown in the following table (Table 5.4). As the values are different for the different levels and s, and S, when they are rounded, the same "rounded" value is found. And after that, it was checked that the first solution and neighbours had a service level $\geq 95\%$.

Table 5.4: Initial s_i and S_i values

		Factory	Distributor	Wholesaler	Retailer
Initial s_i	[pieces]	13000	16000	13000	11000
Initial S_i	[pieces]	13000	16000	13000	11000

5.5 Pseudocode

The notation followed to write the pseudocode is at Table 5.5.

Table 5.5: Notation of the pseudocode II

t	Time step t
Scurrent	Current solution
Scand	Candidate neighbour solution
Sneighbour	Best candidate neighbour solution
Neighbours	All non-forbidden neighbours of a solution
Sbestever	Best solution ever found (S*)

Fbestever	Objective function value of the best solution ever found (F*)
dec	Reducing factor of $\Delta s, \Delta S$ *and* ΔQ, $dec \in [0.75, 0.95]$
Forbidden[i]	Vector with all the 16 moves possible, in each position, there is the time (iterations) until this movement is forbidden.
runs_no_better_found	Counter that saves the number of iterations when no better solutions are found
T	Parameter of the tabu search
Neighbourhood()	Function that computes **Neighbour** given a solution S
Crop()	Function that computes the objective function of a vector of neighbours, and eliminates the ones that do not reach the minimum service level. And after that, eliminates the half worst neighbours.
Run()	Computes the average Cost and Service Level of a solution of a certain amount of iterations.

5.5.1 Main function

Initialization

```
Set Scurrent = Intitial solution
Set Sbestever = Scurrent
Set Fbestever = Objective function value of Sbestever
Set runs_no_better_found = 0;
found = false;
Set i = 0;
while (Time limit not reached)
  • if (runs_no_better_found = 0)
      ○ Set $\Delta Q = \Delta Q_0$
      ○ Set $\Delta s = \Delta s_0$
  • if (runs_no_better_found > T)
      ○ Set $\Delta s = \Delta s * dec$
      ○ Set $\Delta Q = \Delta Q * dec$
      ○ Set runs_no_better_found = 0
  • Call TABU()
  • Set ITER=ITER+1
```

Final updates

```
  • if a better solution is found
      ○ runs_no_better_found = 0;
  • if a better solution is not found,
      ○ runs_no_better_found = runs_no_better_found + 1;
```

5.5.2 TABU function

Initialization

```
Set $\mathrm{Sneighbour = Scurrent}$
Set $\mathrm{ties} = 0$
Set $\mathrm{Bestf} = \infty$
Set $\mathrm{SLmin} = 0.95$
Call **Neighbourhood**()
Call **Crop()**
**While $(\mathrm{i < Neighbours.size})$ do**
    • Set $\mathrm{Scand = Neighbours[i]}$
    • If $\mathrm{(F(Scand) = Bestf)}$
        ○ Set $\mathrm{Ties = Ties + 1}$
        ○ if $\mathrm{random(\,)} \leq \frac{1}{\mathrm{Ties}}$
            ▪ Launch $\mathbf{run}(\mathrm{Costcand_{avg}, SLcand_{avg}})$
            ▪ If$(\mathrm{costcand_{avg} < Bestf\ and\ SLcand_{avg}} \geq 0.95)$
                • Set $\mathrm{Sneighbour = Scand}$
                • Set $\mathrm{Sneighbour.cost = Costcand}$
                • Set $\mathrm{Bestf = Costcand}$
                • Set $\mathrm{Movement_{done} = Sneighbour.mov}$
    • If $\mathrm{(F(Scand) < Bestf)}$
        ○ Launch $\mathbf{run}(\mathrm{Costcand_{avg}, SLcand_{avg}})$
        ○ If$\left(\mathrm{costcand_{avg} < Bestf\ and\ SLcand_{avg}} \geq 0.95\right)$
            ▪ Set $\mathrm{Sneighbour = Scand}$
            ▪ Set $\mathrm{Sneighbour.cost = Costcand}$
            ▪ Set $\mathrm{Bestf = Costcand}$
            ▪ Set $\mathrm{Ties} = 1$
            ▪ Set $\mathrm{Movement_{done} = Sneighbour.mov}$
    • Set $\mathrm{i = i + 1}$
```

Final updates:

```
Set $\mathrm{Scurrent = Sneighbour}$
If $\mathrm{(Bestf < Fbestever)}$
    • $\mathrm{Sbestever = Scurrent}$
    • $\mathrm{Fbestever = Bestf}$
    • Set $\mathrm{found = true}$
    • Set $\mathrm{TAB = Uniform(5,15)}$
```

- $\text{Set time} = \text{TAB} + \text{ITER}$
- $\text{Set Forbidden}[\text{Inverse}(\text{movement})] = \text{time}$

6. Tabu Search, sensitivity analysis

6.1 Parameters of the Tabu search

There are several parameters in the Tabu search, so its values have to be decided on the bases of a sensitivity analysis. As there are different parameters, a great value of combinations can be made. In the Table 6.1 are summarized all the parameters that can be changed in the Tabu search program. The maximum time allowed of the Tabu Search is 1h for the (s,Q) case and 2h for the (s,S) case. As the purpose of this project is academic, the iteration time could be much greater. But if one possible case for this is to let the students use it, it makes more sense to have a reduced time limit. So after observing how and when the curves converge, it is set one hour for the (s,Q) and two hours for the (s,S). Moreover, with these values, several experiments can be pursued in a reduced time space.

Table 6.1: Parameters of the Tabu Search

Parameter	Description
$\Delta \boldsymbol{S}$	Variation parameter for S_i values
$\Delta \boldsymbol{s}$	Variation parameter for s_i values
$\Delta \boldsymbol{Q}$	Variation parameter for Q_i values
Dec. Factor	Decreasing factor of the variation parameters
T	Maximum number of iterations allowed with no better solution found.
TAB	Forbidden number of iterations for a certain movement i. $\mathrm{TAB} = \mathrm{Uniform}(a, b)$

For each case we have a set of $\Delta S/\Delta Q$, Δs, Dec. Factor, T, a and b. In total, there are 6 parameters, so for example, if a set of 2 values per each parameter was implemented, there would be $2^6 = 64$ combinations.

The sensitivity analysis will be performed varying $\Delta S/\Delta Q$, Δs and the decreasing factor. The other values are decided taking into account the space of solutions and number of neighbours that there are.

It is decided to set $a = 5$ and $b = 15$. If at maximum there are 16 neighbours, and in the case that there were tried all of the possible movements; after the 16th iteration there would not be any possible movement if they were forbidden for more than 16 iterations. So it is decided to forbid the movements randomly between 5 and 15 iterations.

The T value is set in 50 iterations. That means that if a solution is not found in 50 iterations, the values are reduced by a decreasing factor. As there are 8 decision variables, if two values were tried, that would be 265 iterations (2^8). If those two values were the minimum and the maximum allowed of each variable, which would mean that the border of the solution space would be evaluated. So the 20% of that would be around 50 iterations. Also, there were tried different T values, 20 and 100; 50 seemed to be the best fit. Even though there were not much differences, since after a certain value the behaviour was similar. Of course, other values could be used and a sensitivity analysis performed, but in this project only $\Delta S/\Delta Q$, Δs and the decreasing factor are analysed.

6.1.1 (s,Q) case

For the reorder point method case (s,Q), the parameters $\Delta Q, \Delta s$ and the decreasing factor are analysed. The variation parameters are 1%, 5% and 10% of the initial solution values; rounding it, the pairs of values are shown on the Table 6.2. The values of the decreasing factor are: $1, 0.95, 0.90, 0.85, 0.80, 0.75$ and 0.70.

Table 6.2: Variation parameters

	1%	5%	10%
$\boldsymbol{\Delta s_0}$	100	500	1000
$\boldsymbol{\Delta Q_0}$	5	25	50

In the following graphs (Figure 6.1, Figure 6.2 and Figure 6.3) [9]there are shown the iteration processes of Tabu search for one run of different combination of parameters.

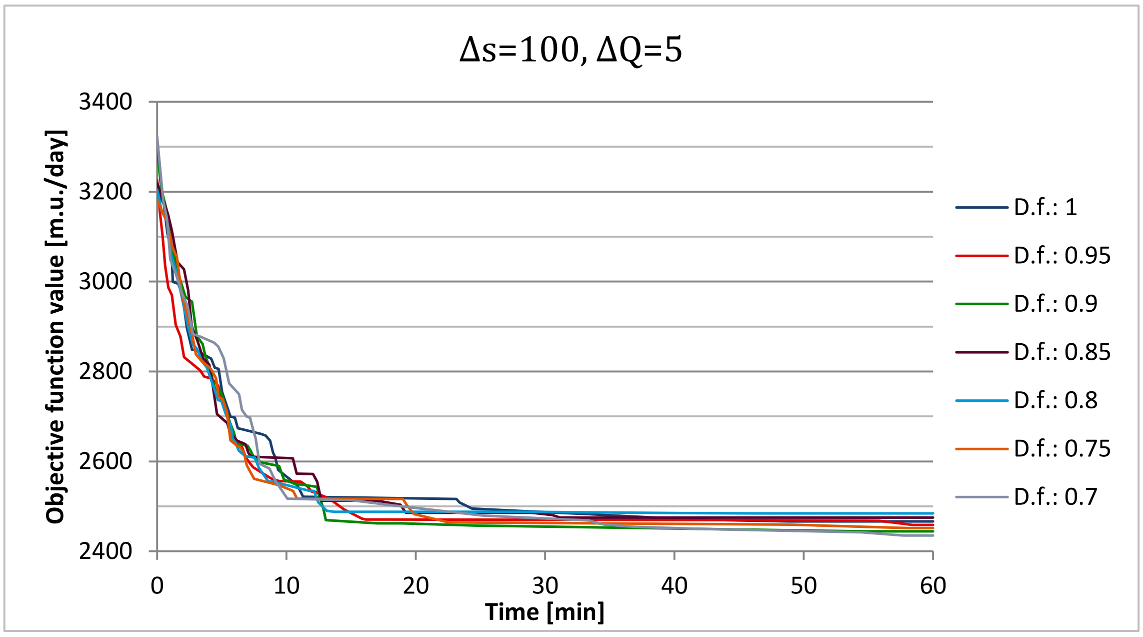


Figure 6.1: Iteration graph of the combinations of D.f.[10] with Δs=100, ΔQ=5

[9] Greater figures can be found in 11.2

[10] D.f.: Decreasing Factor for the variation parameters

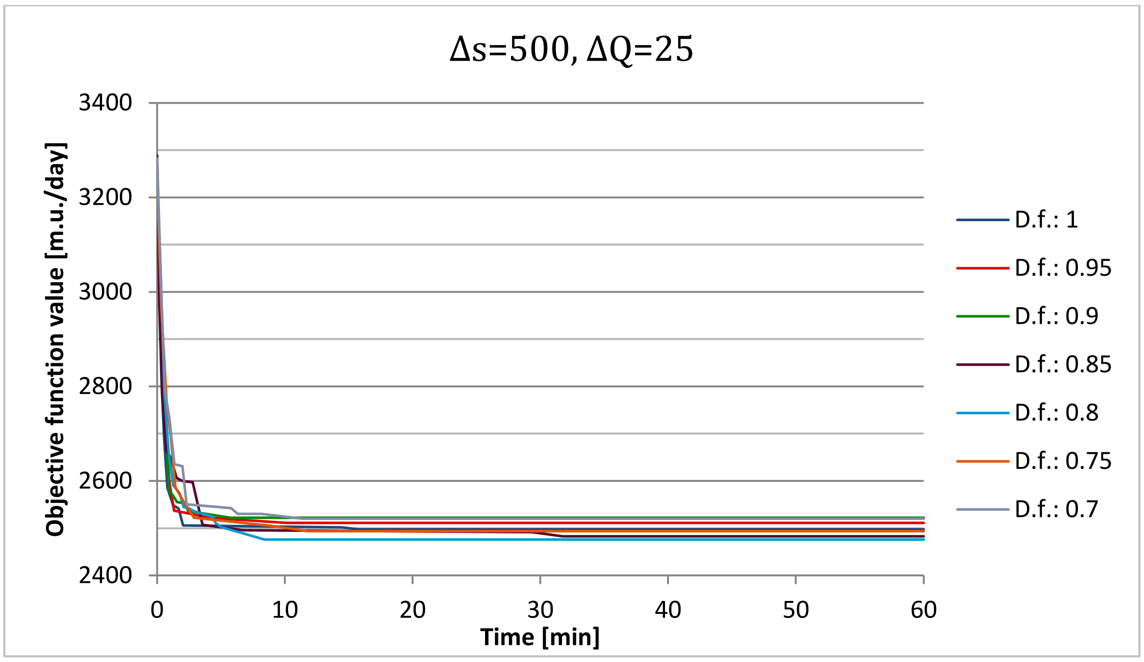


Figure 6.2: Iteration graph of the combinations of D.f. with Δs=500, ΔQ=25

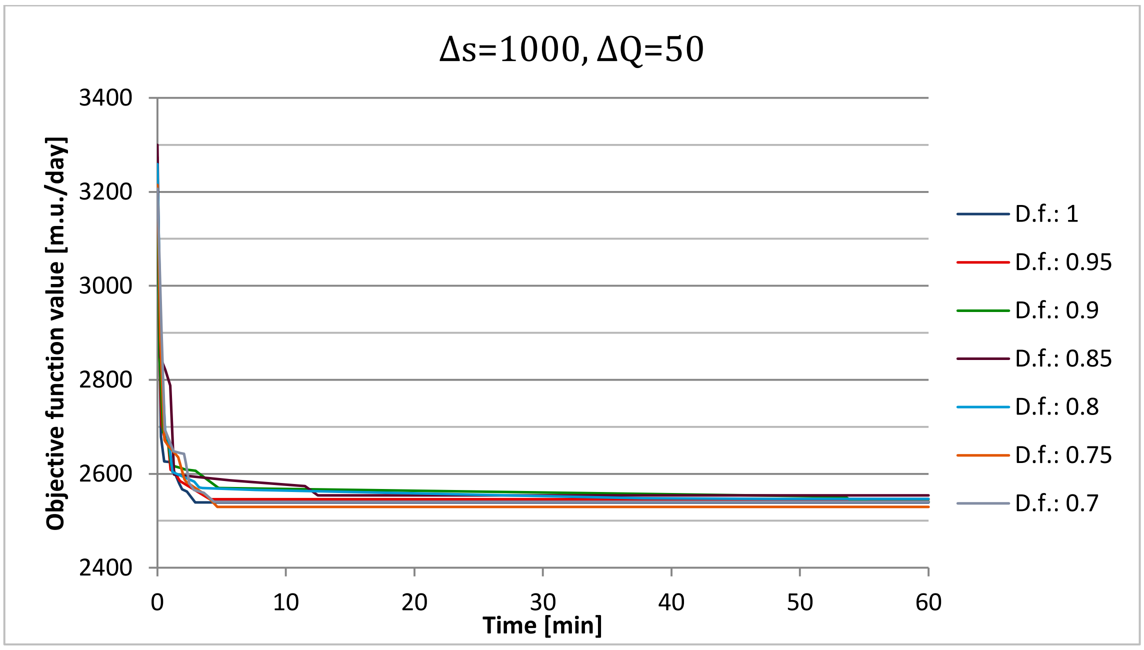


Figure 6.3: Iteration graph of the combinations of D.f. with Δs=1000, ΔQ=50

It can be observed that as the variation parameters increase, better solutions are found faster. In the first 5min, the searches with parameters [Δs=500, ΔQ=25] and [Δs=1000, ΔQ=50] find good solutions, while the search with [Δs=100, ΔQ=5] is still half way there.

But, in contrast, as the variation parameters increase, the best solution found also increases. As the variation parameters increase, the time when they found the same O.F. value is reduced, but the best solution ever found increases.

In the following table, there can be seen the best solution ever found and its time.

Table 6.3: Best solution found

		1	0.95	0.9	0.85	0.8	0.75	0.7
Δs=100	ΔQ=5	2466.59	2458.83	2444.14	2475.11	2484.09	2451.64	2435.01
Δs=500	ΔQ=25	2497.92	2511.32	2522.08	2482.76	2476.05	2493.65	2520.13
Δs=1000	ΔQ=50	2539.19	2546.14	2540.81	2554.05	2546.71	2529.50	2540.61

Table 6.4: Time [s] when the best solution was found

		1	0.95	0.9	0.85	0.8	0.75	0.7
Δs=100	ΔQ=5	2931.11	3504.65	3280.35	1869.06	3179.45	3491.27	3462.20
Δs=500	ΔQ=25	946.21	605.70	344.88	1904.99	503.40	706.38	684.35
Δs=1000	ΔQ=50	176.28	245.90	3323.13	748.32	2868.67	279.97	265.35

6.1.1.1 Validation

A validation of the Tabu Search has to be performed. Also, it is needed to validate the dynamic search with respect a static search. To validate the Tabu search it is used a Descent Local Search (DLS). For this method it is used the same code as for the Tabu Search with some differences. All moves are allowed in all steps, the initial solution is the initial solution of the tab search with a perturbation applied at all components of the vector. The perturbation is a random value $z \in [0.75, 1.5]$. After a certain amount of runs (50) without finding any better solution, the search restarts with a different initial solution, until the end time is reached.

The validation of the dynamic tabu search is performed with a static tabu search, with all the parameters as the dynamic tabu search but the decreasing factor. The decreasing factor is 1 as it is a static search.

Table 6.5: Objective function value of the best solution found at each run

Dynamic TS	Static TS	DLS
F* [m.u.]	F* [m.u.]	F* [m.u.]
2511.6	2465.7	2501.5
2500.0	2491.9	2750.1
2473.6	2403.6	2568.5
2473.3	2451.8	2529.9
2475.6	2432.6	2786.9
2445.7	2431.2	3158.6
2474.4	2441.1	2821.7
2482.4	2427.9	2497.1
2462.7	2436.7	3222.3
2473.3	2435.5	2492.3

Table 6.6: Time passed until the best solution is found at each run

Dynamic TS	Static TS	DLS
t* [s]	t* [s]	t* [s]
2323.9	2754.9	1829.3
81.8	2226.4	2859.5
709.3	2728.8	3018.3
3531.3	3274.0	435.0
422.0	3489.5	1791.8
1905.6	2905.6	1253.4
1330.3	3401.8	1357.7
1253.7	3147.6	313.8
2361.1	1833.7	0.0
3531.3	2700.6	1407.5

On the previous tables (Table 6.5 and Table 6.6) the values of the several runs are shown, the best solution found, and the time passed until it was found.

In the following graphs, to observe better the difference among the three programs, a boxplot is made. So all the values found on the previous tables are represented using quartiles. Also, the confidence interval of the 95% for the mean is shown as well.

One of the better combinations is Δs=500 and ΔQ=25 with a decreasing factor of 0.8. So those will be the parameters for the DLS and TS static. So the three methods will be compared in the following graphs, showing the best solution found, and the time when it was found.

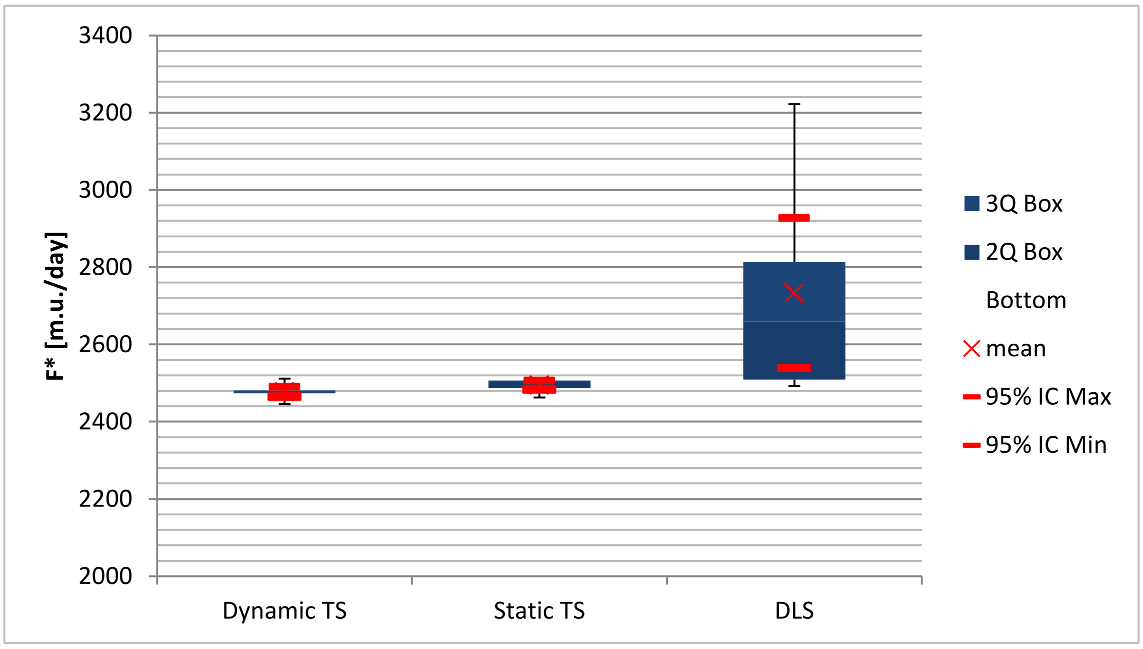


Figure 6.4: Boxplot of the best solution found in 10 runs for DTS, STS and DLS

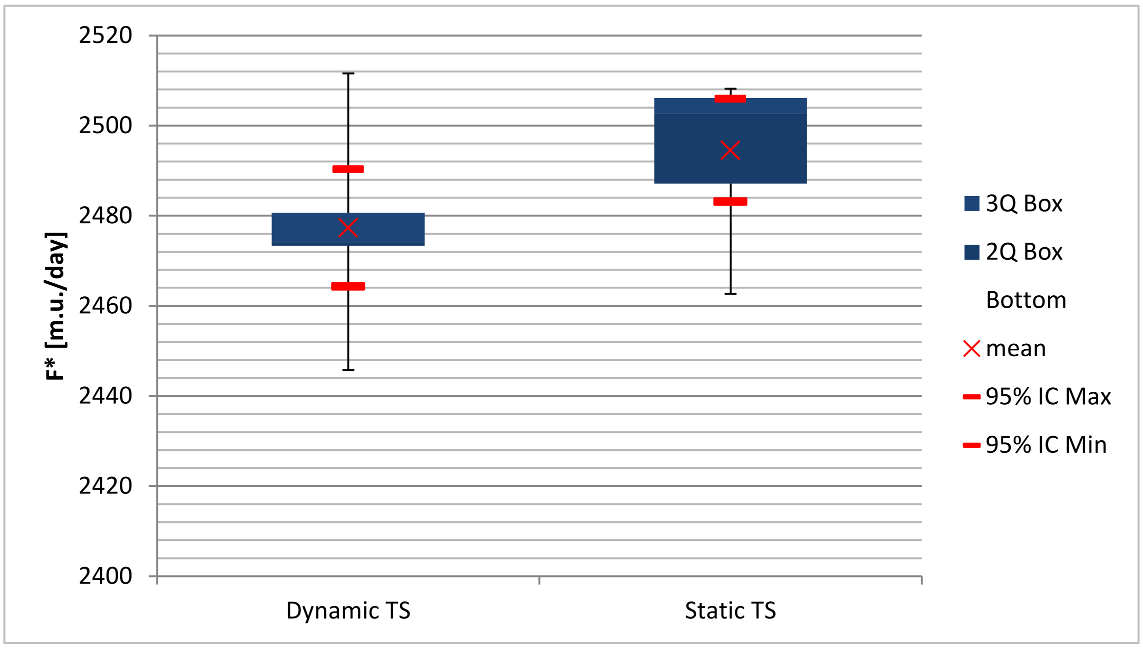


Figure 6.5: Boxplot of the best solution found in 10 runs for DTS and STS

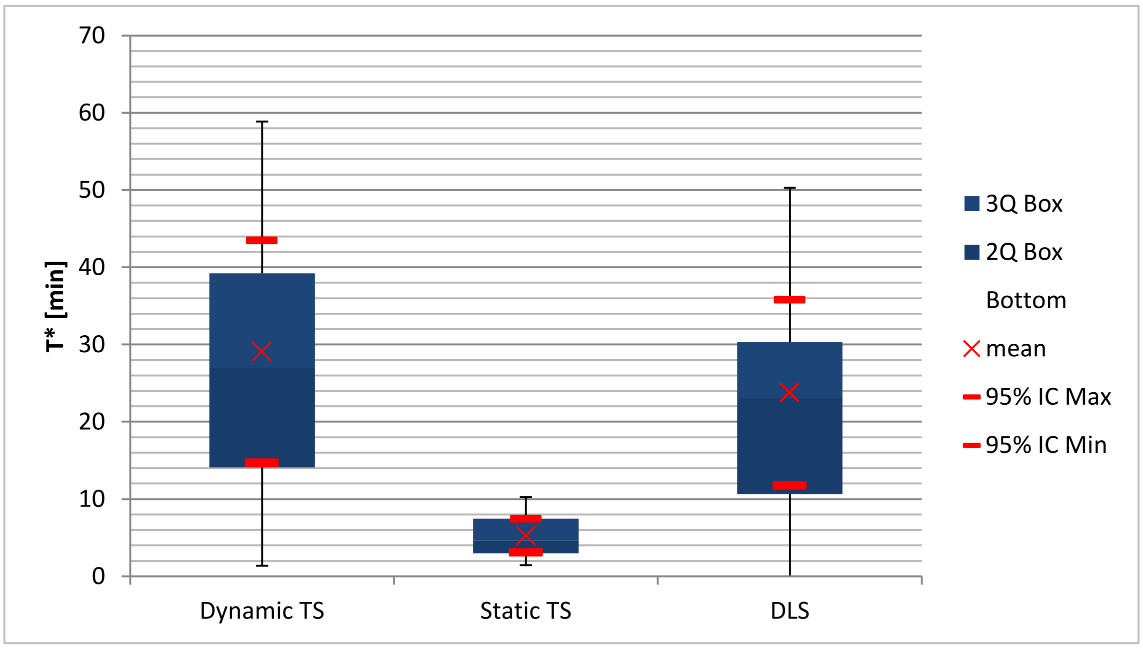


Figure 6.6: Boxplot of the time when the best solution was found in 10 runs for DTS, STS and DLS

In the graphs it can be seen that the descent local search does not perform very well, its time is fast but the solutions are far the best solution found using the tabu methods.

It can be concluded that for this case, even though the Static TS is faster, the dynamic tabu search finds better solutions than the STS. So the dynamic tabu search performs better for Δs=500 and ΔQ=25 than the static tabu search.

6.1.2 (s,S) case

For the reorder point method case (s,S), the parameters $\Delta S, \Delta s$ and the decreasing factor are analysed. The variation parameters are 1%, 5% and 10% of the initial solution values; rounding it, the pairs of values are shown on the Table 6.2. The values of the decreasing factor are: $1, 0.95, 0.90, 0.85, 0.80, 0.75$ and 0.70. After running the program several times, it is decided a duration of 2h, observing he graphs it is deducted that 1h is not enough.

Table 6.7: Variation parameters

	1%	5%	10%
Δs_0	100	500	1000
ΔS_0	100	500	1000

In the following graphs (Figure 6.7, Figure 6.8 and Figure 6.9) are shown the iterations for combinations Δs, ΔS and several decreasing factor, as in the (s,Q) method.

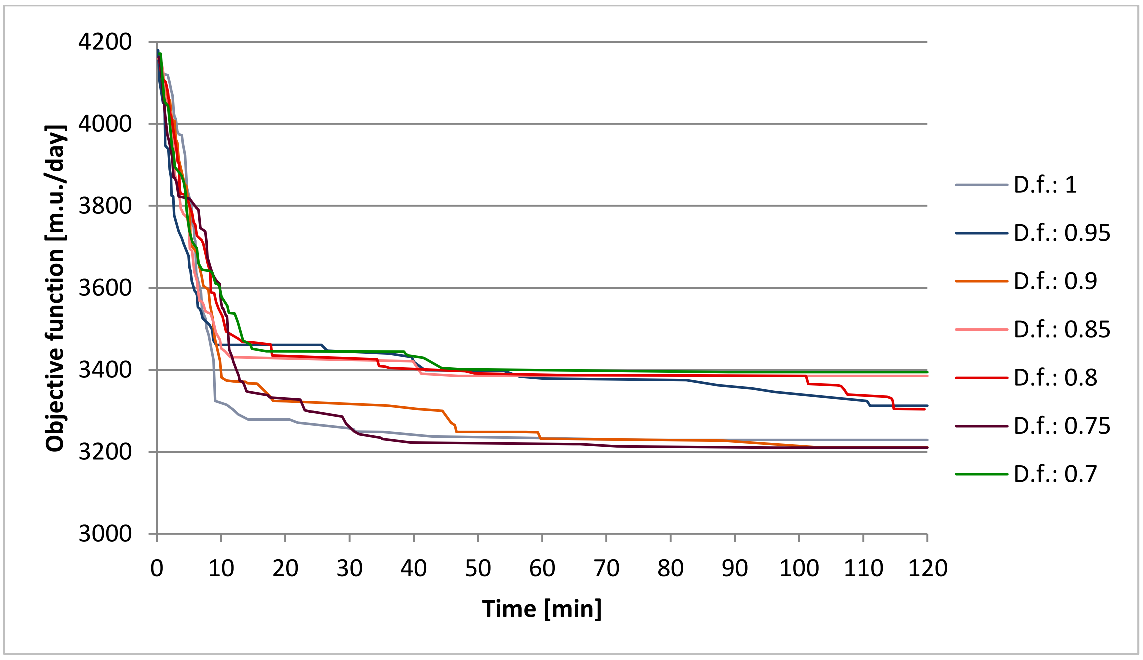


Figure 6.7: Iteration graph of the combinations of D.f. with Δs=100, ΔS=100

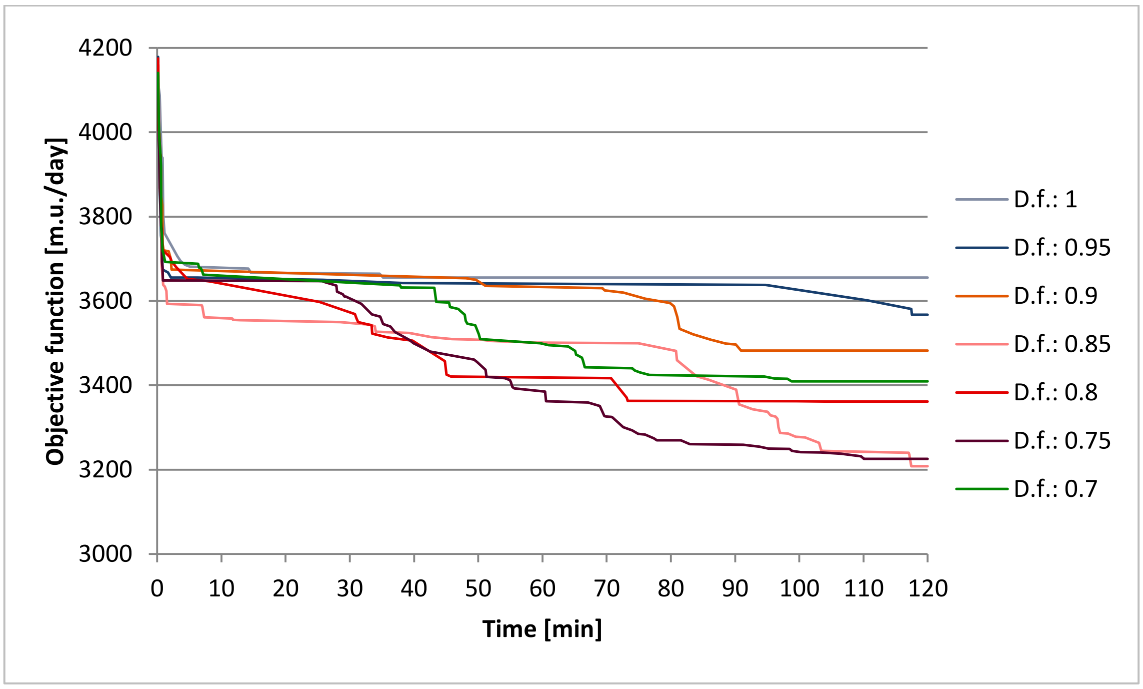


Figure 6.8: Iteration graph of the combinations of D.f. with Δs=500, ΔS=500

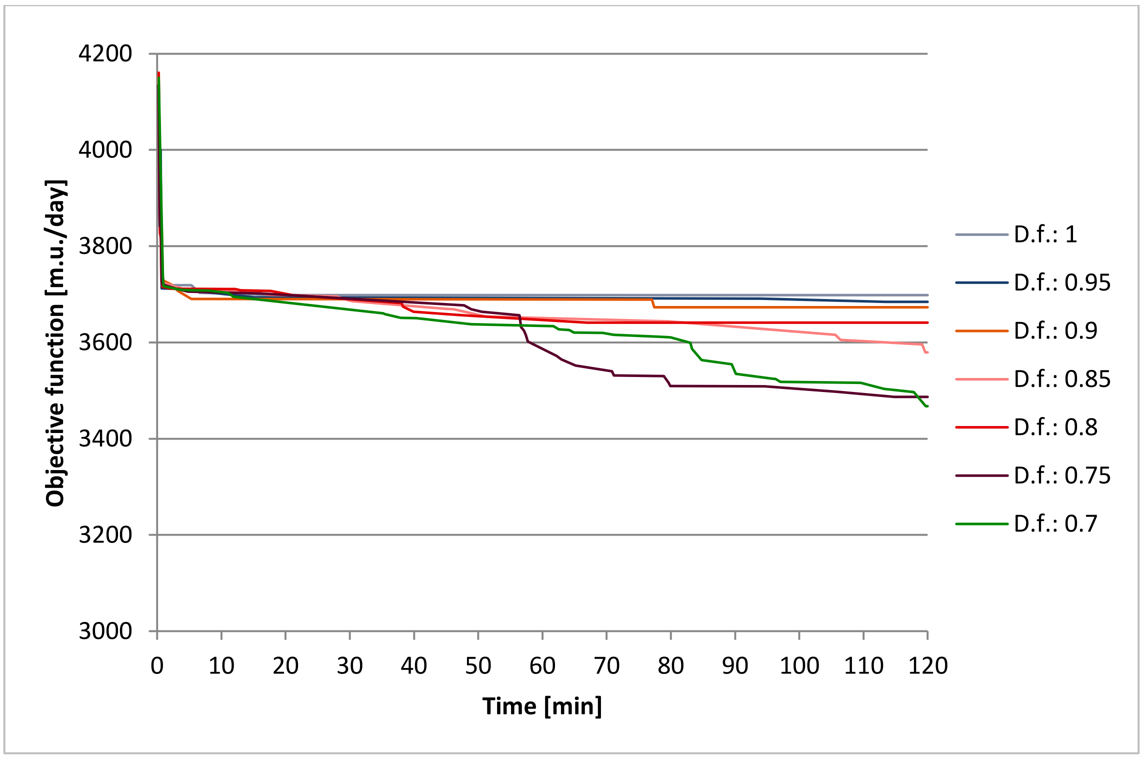


Figure 6.9: Iteration graph of the combinations of D.f. with Δs=1000, ΔS=1000

On those graphs it can be observed that for great variations: 500 and 1000, at the beginning descent quicker than for the 100 variation. But after that huge descent almost all the curves go flat. That does not happen in the others graphs (Figure 6.8 and Figure 6.9) as after a while (30 min for the first and 50 for the second) they have an important descent towards the best solution.

In this method (s,S) the decreasing factors have more impact as the variation parameters increase. In the first graph (Figure 6.7) there is no such correlation, but in the others (Figure 6.8 and Figure 6.9) the value of the best solution is almost sorted as the decreasing factor. That can be observed very well in Figure 6.9, where all except D.f =0.85 and 0.9 are sorted exactly the same as the best solution found.

To sum up, a table is made to show in numbers what it is on the graphs.

Table 6.8: Objective function value of the best solution found II

		1	0.95	0.9	0.85	0.8	0.75	0.7
Δs=100	ΔS=100	3229.12	3311.98	3210.94	3385.22	3303.31	3210.37	3394.29
Δs=500	ΔS=500	3655.59	3567.57	3482.33	3208.19	3361.08	3225.61	3409.01
Δs=1000	ΔS=1000	3698.2	3684.42	3672.86	3579.33	3640.99	3487.16	3467.79

Table 6.9: Time [s] when the best solution was found

		1	0.95	0.9	0.85	0.8	0.75	0.7
Δs=100	ΔS=100	4566.82	6662.46	6175.19	2813.79	7172.53	5762.40	5337.97
Δs=500	ΔS=500	2112.47	7052.35	5455.32	7046.10	6247.64	6603.49	5927.50
Δs=1000	ΔS=1000	1089.44	6806.67	4642.94	7178.22	4015.96	6887.45	7181.02

6.1.2.1 Validation

This method requires a validation as well. So as explained in the previous chapter, a Descent Local Search and a Static Tabu search are performed. The parameters will be Δs=100 and ΔS=100, as this combined with 0.75 of decreasing factor in the Dynamic Tabu Search performs very well.

Table 6.10: Objective function value of the best solution found at each run II

Dynamic TS	Static TS	DLS
F* [m.u.]	F* [m.u.]	F* [m.u.]
3264.0	3212.2	3643.2
3224.5	3224.8	6820.4
3339.8	3272.4	3536.2
3302.9	3364.3	5487.1
3230.3	3367.2	4908.5
3327.6	3232.8	3539.7
3268.4	3278.5	3603.6
3199.0	3239.4	3696.9
3299.2	3323.1	4496.5
3319.6	3313.4	4165.3

Table 6.11: Time passed until the best solution is found at each run II

Dynamic TS	Static TS	DLS
t* [s]	t* [s]	t* [s]
6936.0	2819.2	1198.3
6074.0	2551.3	2094.5
5698.2	4004.3	260.9
7154.5	6772.0	3490.5
6252.1	3021.4	5334.8
6829.0	6775.2	1011.8
6977.1	6986.8	366.7
4285.2	5731.8	584.5
6190.6	6342.3	1279.0
7105.6	6785.8	2632.2

As on the previous chapter, the values of the runs can be found on a table (Table 6.10 and Table 6.11). The values are the objective function value of the best solution found at each run, and the time when it is found.

In order to compare in a more visual way the values, two boxplots are made, one with the best values found in the 10 runs and the second one with the time when the best solution is found.

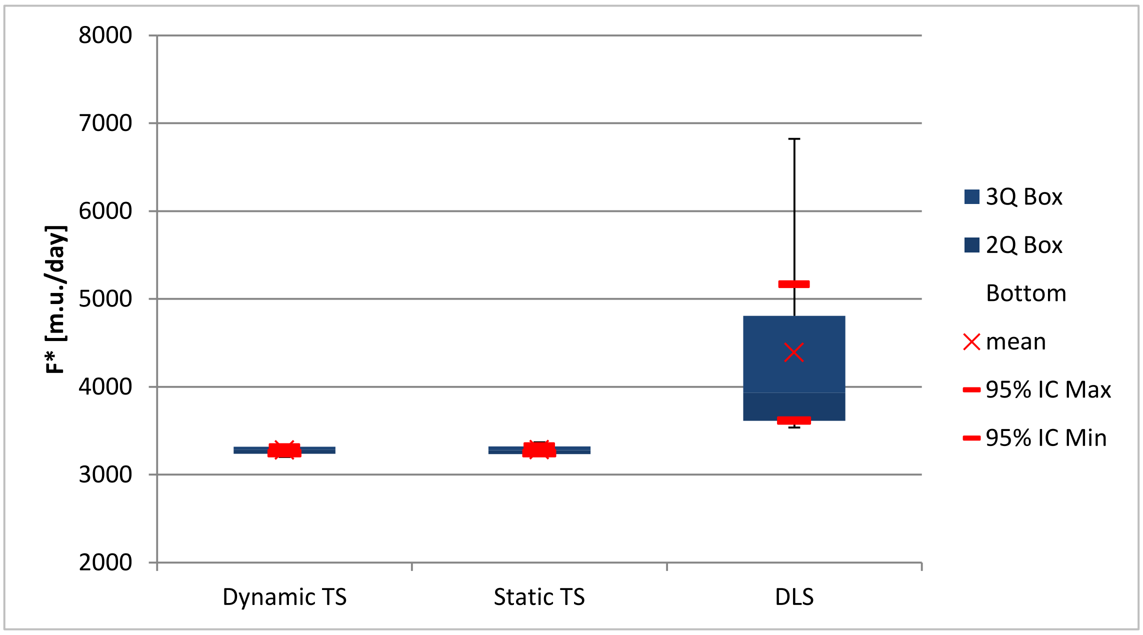


Figure 6.10: Boxplot of the best solution found in 10 runs for DTS, STS and DLS

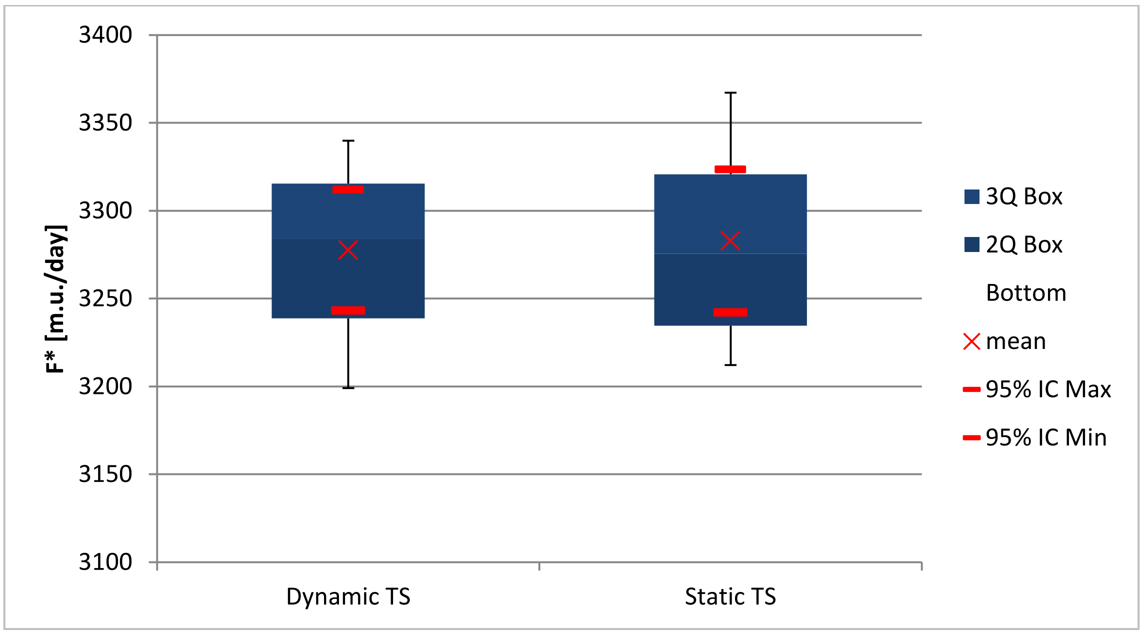


Figure 6.11: Boxplot of the best solution found in 10 runs for DTS and STS

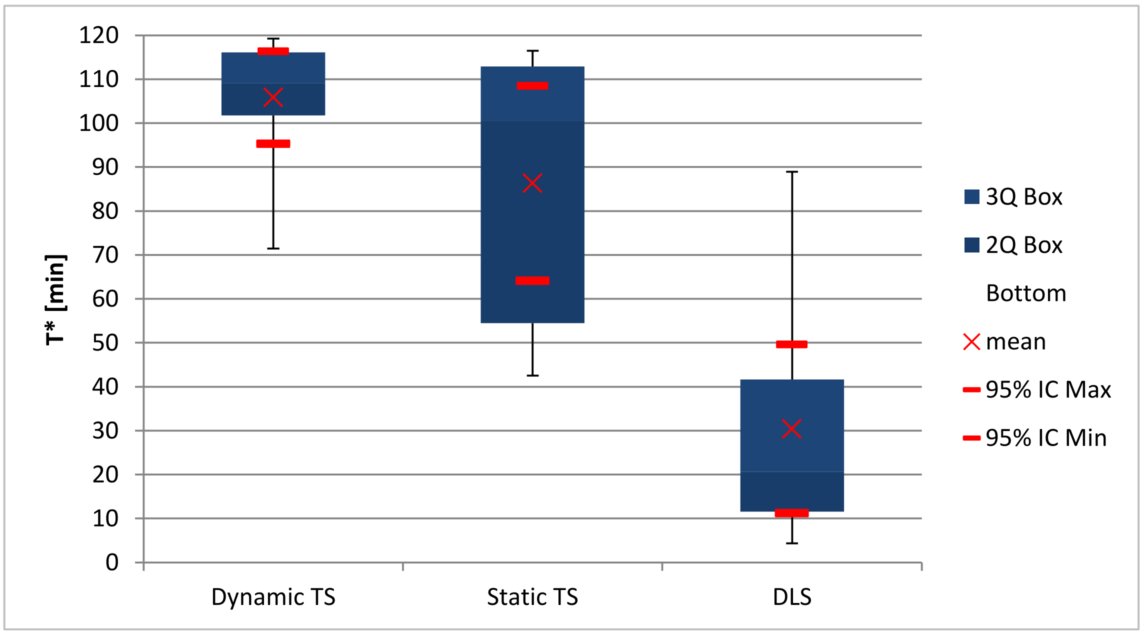


Figure 6.12: Boxplot of the time when the best solution was found in 10 runs for DTS, STS and DLS

The Descent Local search does not perform as the tabu search; the solutions found are far greater than the found by the two tabu methods.

The dynamic tabu search performs better than the static, for this pair of Δs and ΔS the confidence interval for the mean is smaller. Also the extremes of the confidence interval of a 95% are a bit below the values of the static tabu search.

6. Tabu Search, sensitivity analysis

The time when the dynamic tabu search founds the best solution is greater than with the Static Tabu search. That happens because the dynamic tabu search keeps decreasing the variation parameter with the factor 0.75 until it founds a better solution.

There is not as much difference as in the (s,Q) case because the variation parameters $\Delta s=100$ and $\Delta S=100$ are not as great as in the (s,Q). So the decreasing values do not affect as much.

7.Results

(s,Q) case best solution

Using the dynamic tabu search, the best solution is found and shown in the following table: 7.1

Table 7.1: Solution found for (s,Q) case

		Retailer	Wholesaler	Distributor	Factory
s_i	[pieces]	10071	12275	15204	11884
Q_i	[pieces]	420	560	644	966

The confidence interval of 95% for the daily cost is $[2372.56, 2402.41]$ and the confidence interval for the service level is: $[0.937, 0.957]$. In the following graphs there can be observed the inventory level and the on-hand stock at the end of each period for all the echelons of one simulation. In order to be able to appreciate the peaks, only 100 days are shown. It is important to remind that each day has 10 steps, so between 0 and 10 days, there are 100 steps.

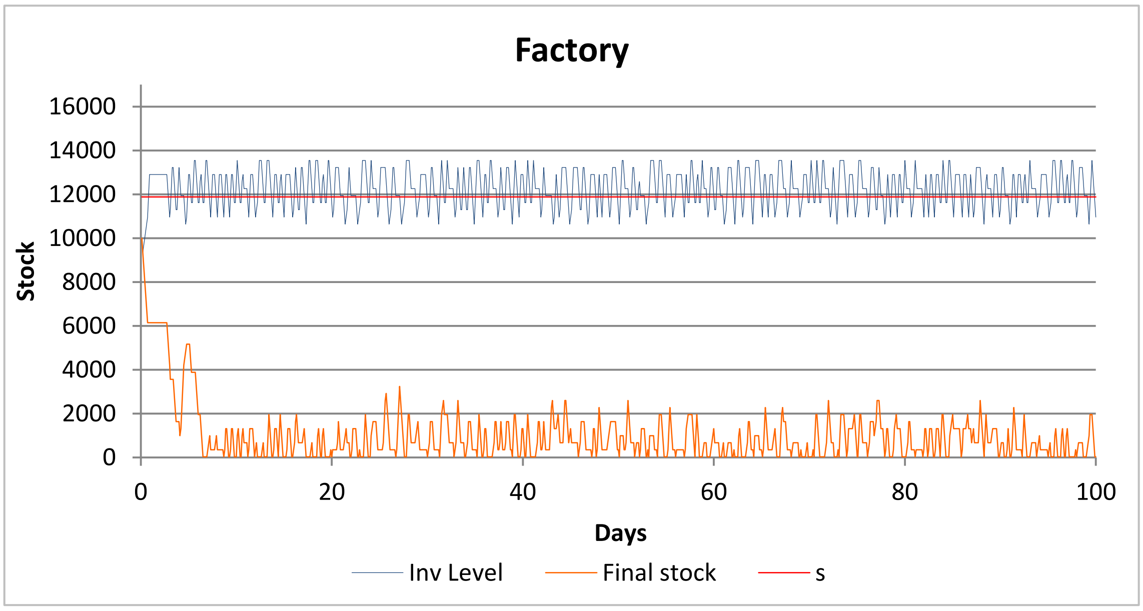


Figure 7.1: Inventory level, on-hand stock at the end of each period, and reorder point for the Factory

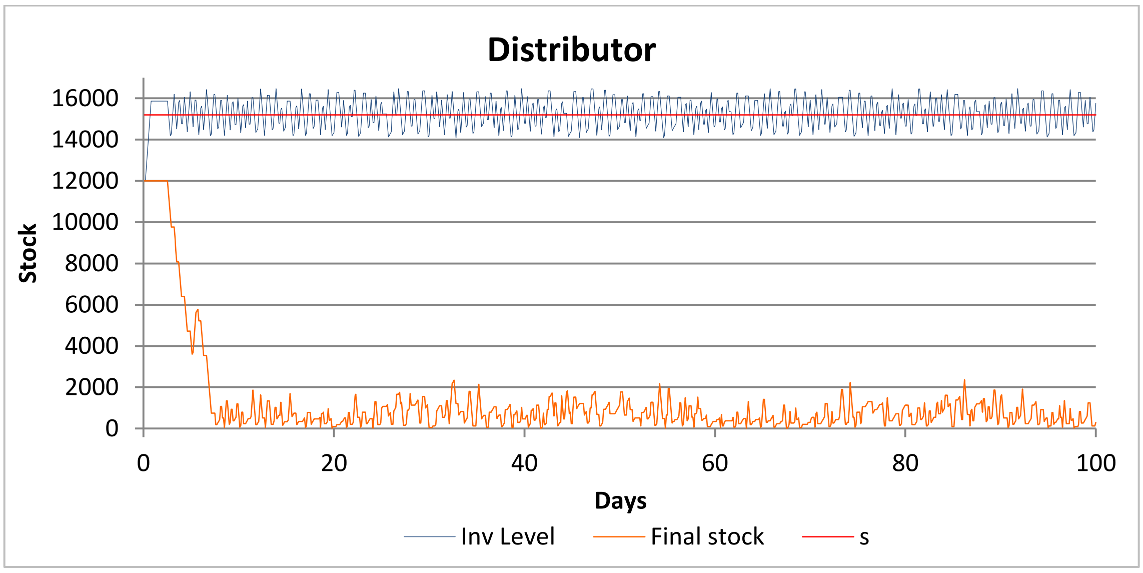


Figure 7.2: Inventory level, on-hand stock at the end of each period, and reorder point for the Distributor

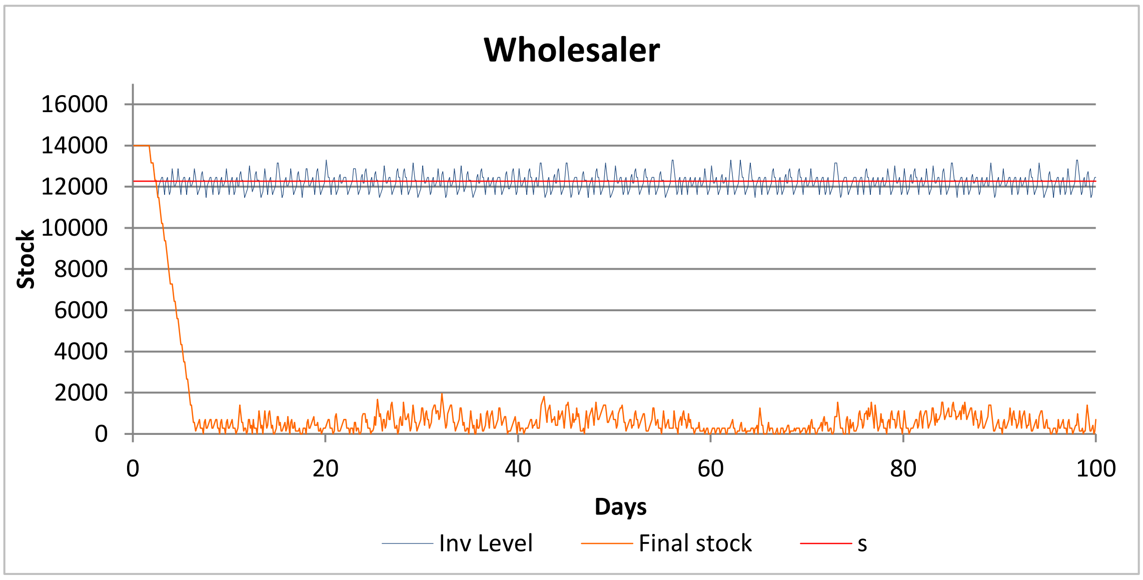


Figure 7.3: Inventory level, on-hand stock at the end of each period, and reorder point for the Wholesaler

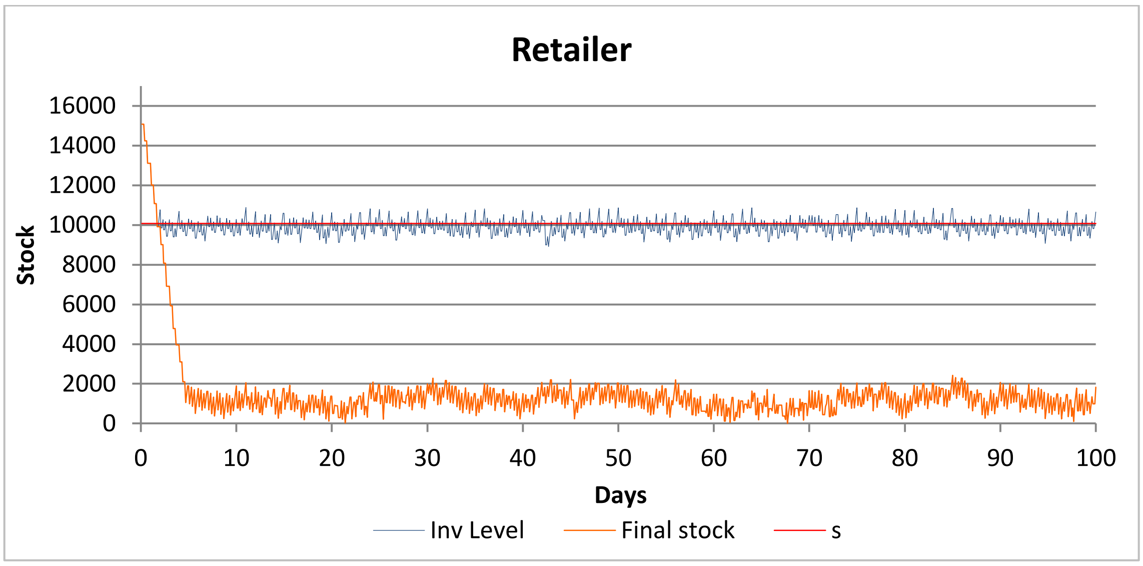


Figure 7.4: Inventory level, on-hand stock at the end of each period, and reorder point for the Retailer

It can be observed that the echelons order more frequently in a downstream sense. That is because the order quantity decreases as it goes downstream.

The average on-hand stock at the end of the period is shown in the following graphs. The total on-hand stock average is 3291 pieces, more than the mean of the daily demand.

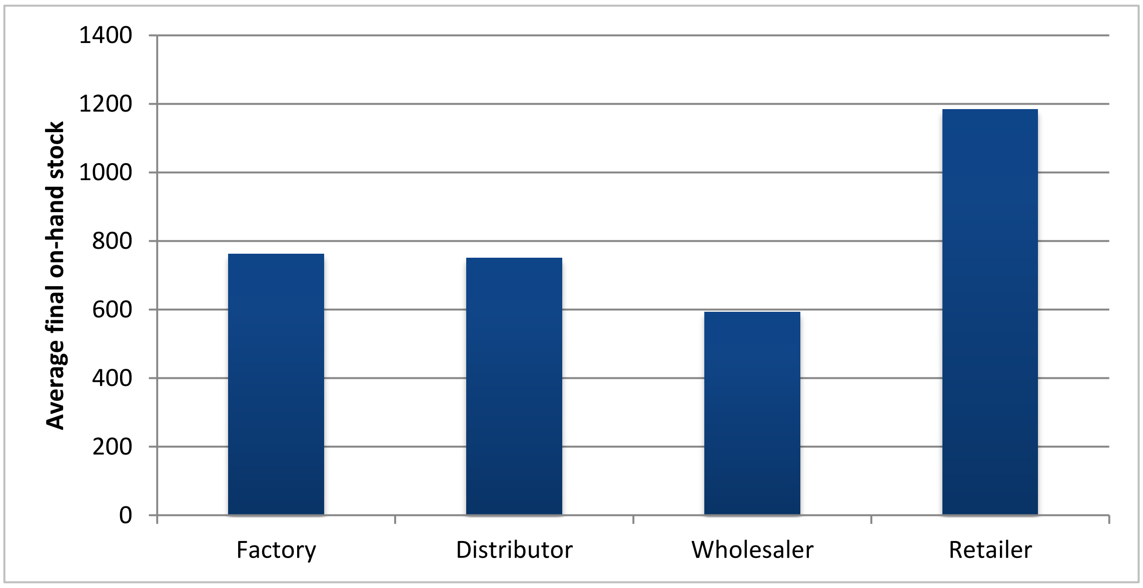


Figure 7.5: Average on-hand stock at the end of the period of 500 days for each echelon

As can be seen on the previous figure, the on-hand stock at the retailers is the greatest one, the stock increases from the wholesaler to the distributor. Some factors can be the decreasing value of the holding cost. The factory does not have that great on-hand stock probably caused by the small holding cost and the reduced lead time.

7.1.1 Results for several service levels

It is interesting also to know how could vary the solution and the cost if the service level was lower and higher than 95%. So two searches were launched to compute the best solution for the case SL=0.90 and SL=0.98. As the case is stochastic, even though a solution was found for a SL=0.98, when the solution was simulated 20 times, the average service level was 0.97.

The solution found are shown in the following table (Table 7.2), the solution for a 95% service level is the one found in the previous chapter.

Table 7.2: Best solution for service levels: 90%, 95% and 97%

	s_i				Q_i			
	F	D	W	R	F	D	W	R
SL=90%	11614	15261	12327	10000	926	579	548	411
SL=95%	11884	15204	12275	10071	966	644	560	420
SL=97%	11976	15200	11992	10592	916	615	539	361

In the following graphs (Figure 7.6) are represented the values on the table. In the reorder point level a slightly tendency can be observed. But it cannot be assured that it is caused by the service level, as it could possible happen that another fitting solution could show a different tendency.

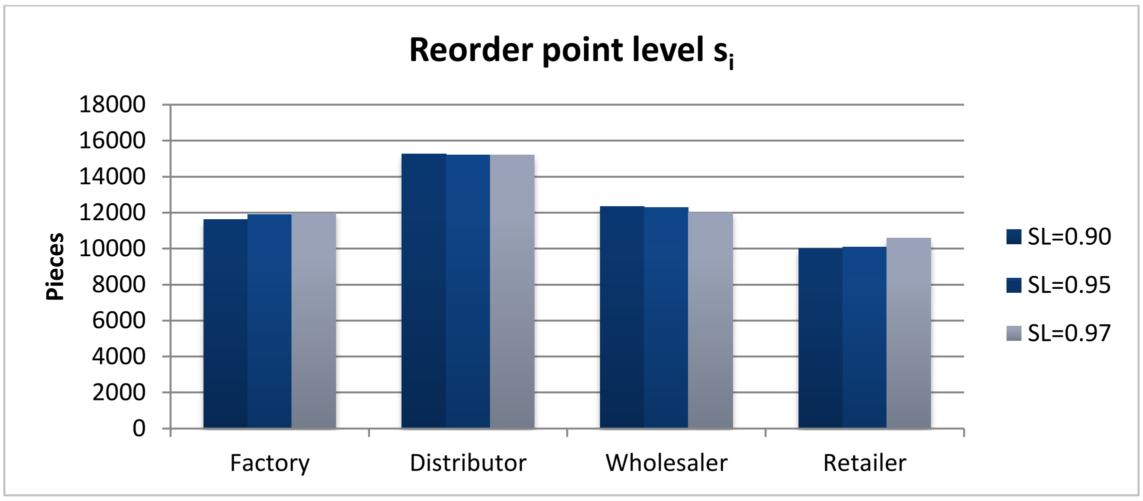


Figure 7.6: Reorder point levels for the service levels: 90%, 95% and 97%

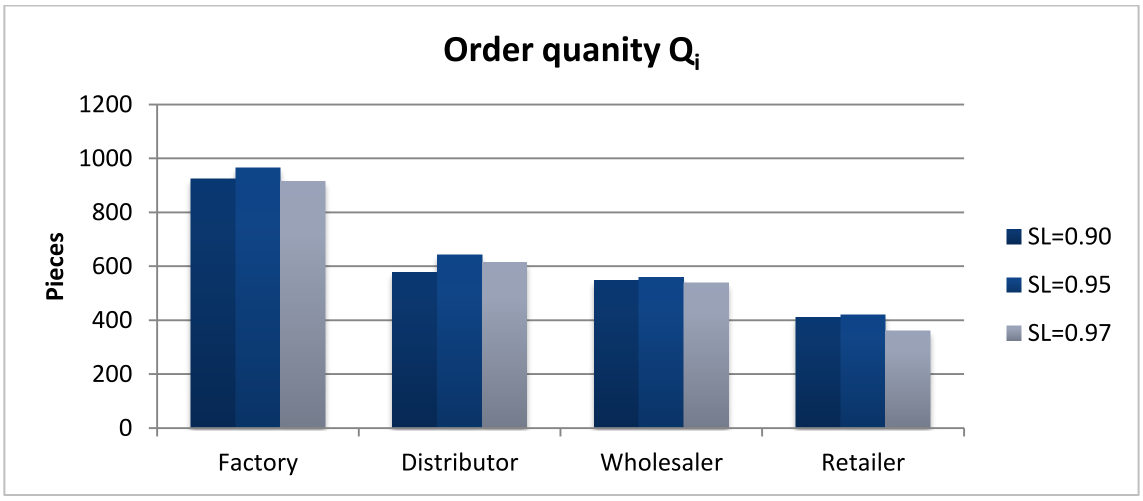


Figure 7.7: Order quantity for the service levels: 90%, 95% and 97%

It can be observed in the following graph that there is an important difference regarding the average on-hand stock at the retailer's at it increases as the service level increases.

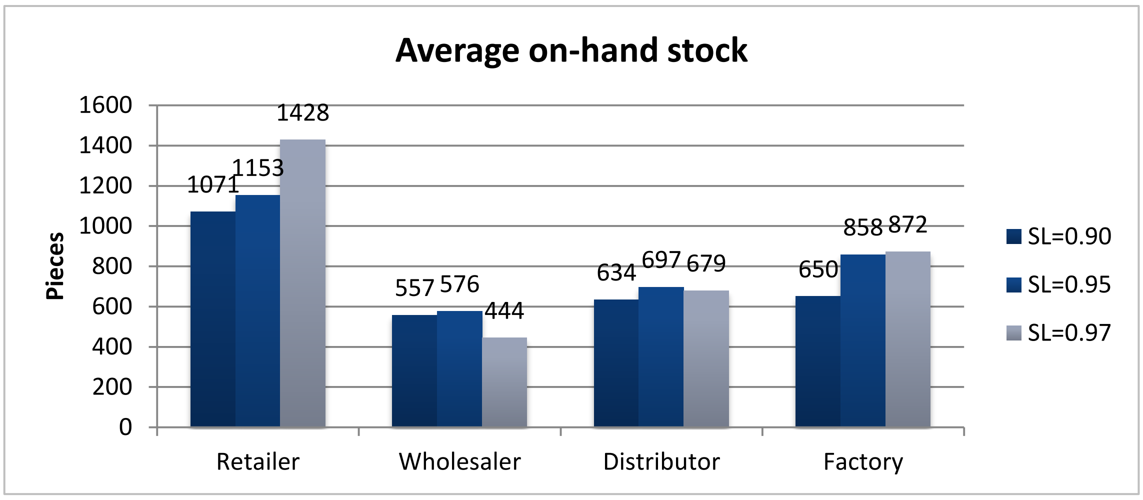


Figure 7.8: Average on-hand stock of the simulation using the solutions found for SL=90%, 95% and 97%

Ten simulations were run to enclose the daily cost for the different service levels. The intervals of confidence for the mean of the daily cost were computed (Table 7.3). It can be noticed that they do not overlap. So it can be said that in 95% of cases the average daily cost of SL=90% will be smaller than the daily cost for a SL=95%, and also, the daily cost of SL=95% will be smaller than the daily cost of the solution of SL=97%.

Table 7.3: 95% confidence interval for the mean of the daily cost

	95% Confidence interval for the mean
$\boldsymbol{Service\ level = 90\%}$	$\{2329.50, 2368.94\}$
$\boldsymbol{Service\ level = 95\%}$	$\{2387.55\ ,2433.03\}$
$\boldsymbol{Service\ level = 97\%}$	$\{2604.03, 2655.93\}$

In the following graph (Figure 7.9) it is shown the values in a boxplot of the 10 runs for each solution of service level. In the 2Q box are shown the second quartile and in the 3Q box the third quartile. The red cross is the mean, and it can be seen that the mean does not follow a straight line, so the daily cost is not linearly proportional to the service level.

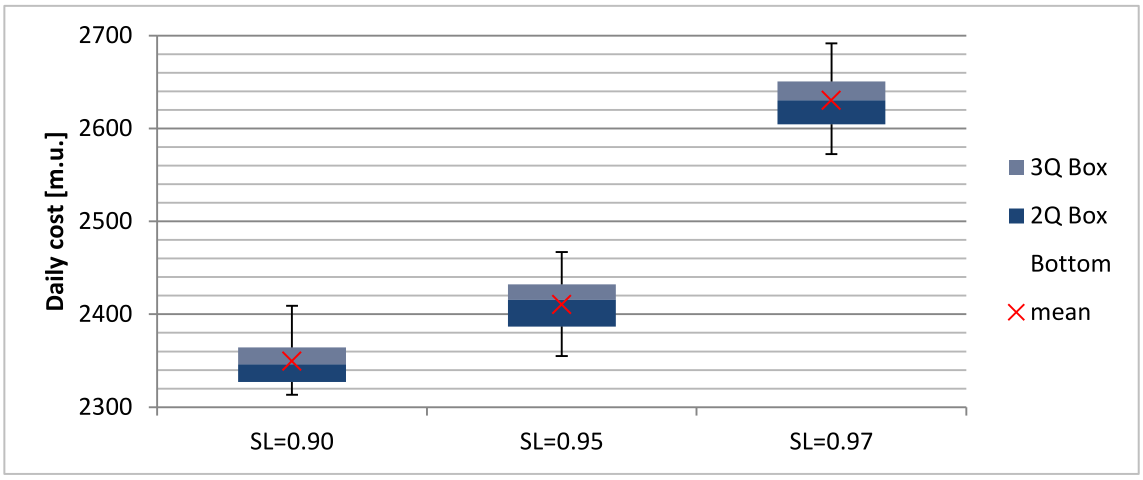


Figure 7.9: Boxplot of the runs of the solutions for service levels = 90%, 95% and 97%

7.2 (s,S) case best solution

The solutions for the (s,S) case are shown in the following table (Table 7.4). It can be observed that the levels are really close, that shows that it is rather closer to a base stock (only one level) policy than to an (s,S) policy. So this model could be simplified to one when each period is ordered the difference between S and the inventory position, and not only when the inventory position is smaller than s.

Table 7.4 Best solution found for (s,S) case

		Factory	Distributor	Wholesaler	Retailer
s_i	[pieces]	12056	15940	12163	10419
S_i	[pieces]	12124	16090	12178	10430

In the following graphs there are shown the inventory position, s level and on-hand stock at the end of each period. It can be notice that as more up-stream, the peaks of the inventory position increase. And that results in greater on-hand stock.

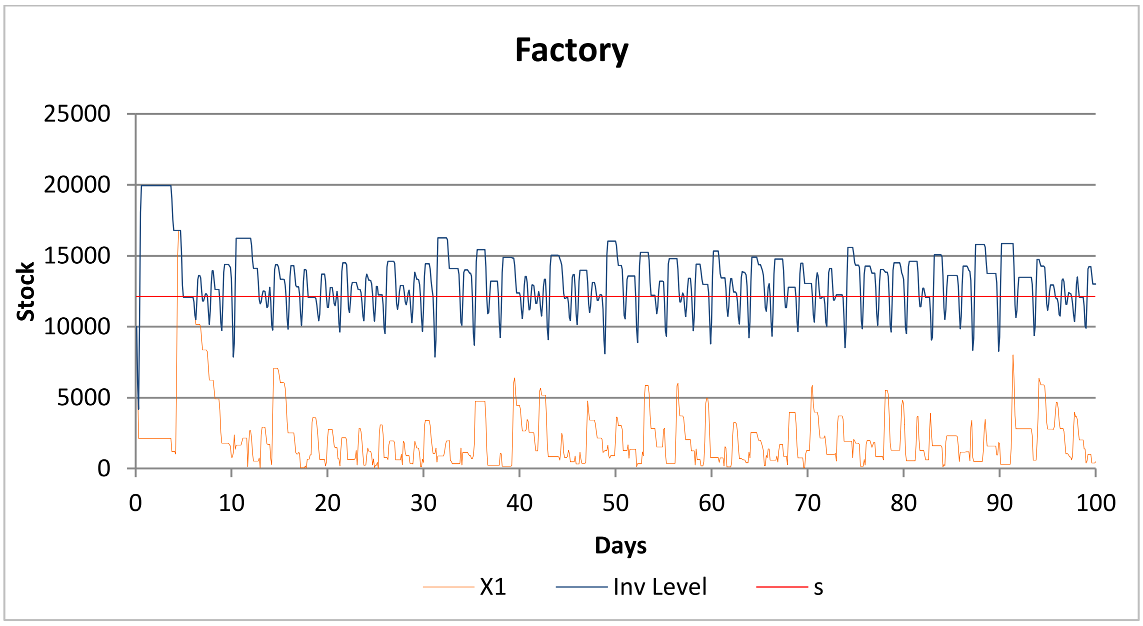


Figure 7.10: Inventory level, on-hand stock at the end of each period, and reorder point for the Factory

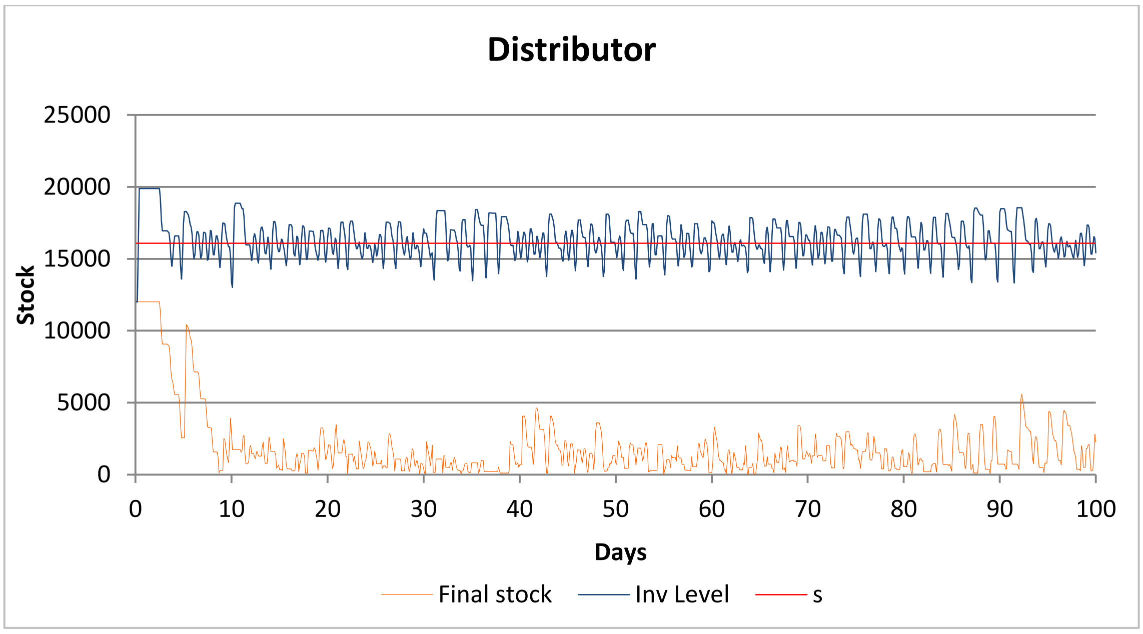


Figure 7.11: Inventory level, on-hand stock at the end of each period, and reorder point for the Distributor

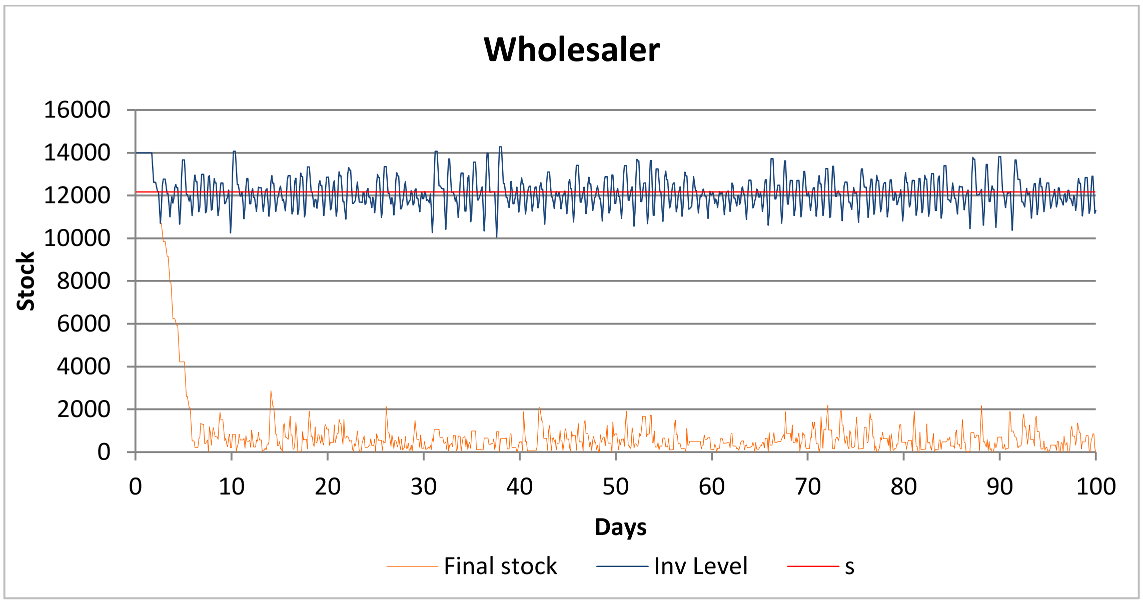


Figure 7.12: Inventory level, on-hand stock at the end of each period, and reorder point for the Wholesaler

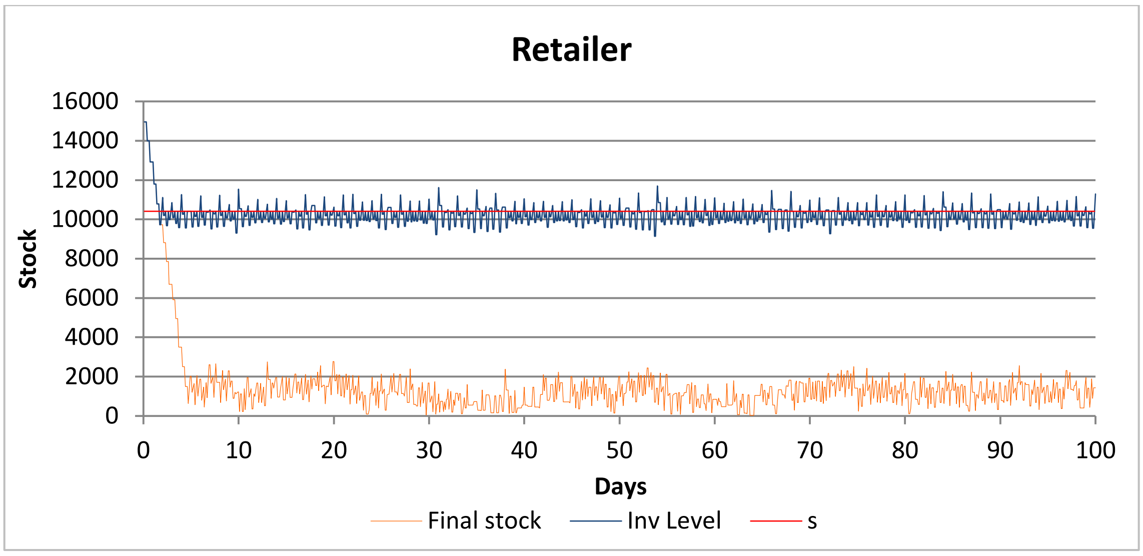


Figure 7.13: Inventory level, on-hand stock at the end of each period, and reorder point for the Retailer

A graph representing the average on-hand stock is made, in this graph (Figure 7.14) the quantity is shown, and it shows what appears in the previous graphs, the on-hand stock is accumulated at the distributor's and at the factory's. The addition of the average on-hand stock is: 5266 pieces.

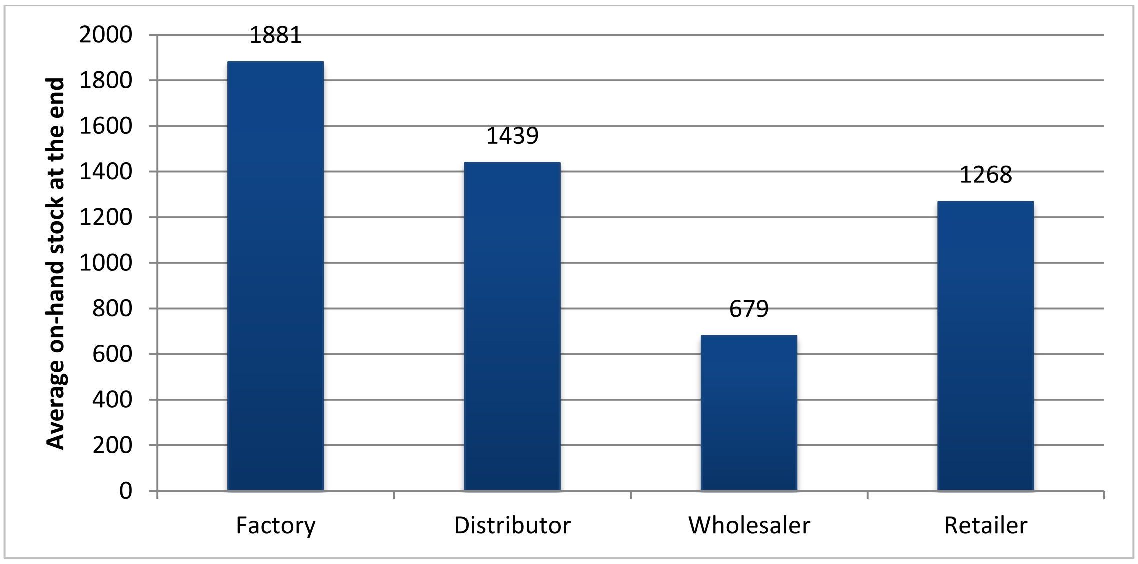


Figure 7.14: Average on-hand stock at the end of the period of 500 days for each echelon II

7.2.1 Results for several service levels

As done with the (s,Q) case, the comparison between different service levels is done as well for the (s,S) case. The service levels are SL=0.88, 0.94 and 0.97. A table with the solutions is made (Table 7.5).

Table 7.5: Best solution for service levels: 88%, 94% and 97%

	s_i				S_i			
	F	D	W	R	F	D	W	R
SL=88%	11891	15768	12039	10345	15815	12142	10345	11891
SL=94%	12056	15940	12163	10419	16090	12178	10430	12056
SL=97%	12619	15195	12884	10498	15220	12896	10555	12619

The values of the table are represented on the two following figures (Figure 7.15 and Figure 7.16). It can be observed a slight tendency but it is not conclusive because not all the fitting solutions and for all service levels between 0.88 and 0.97 are computed.

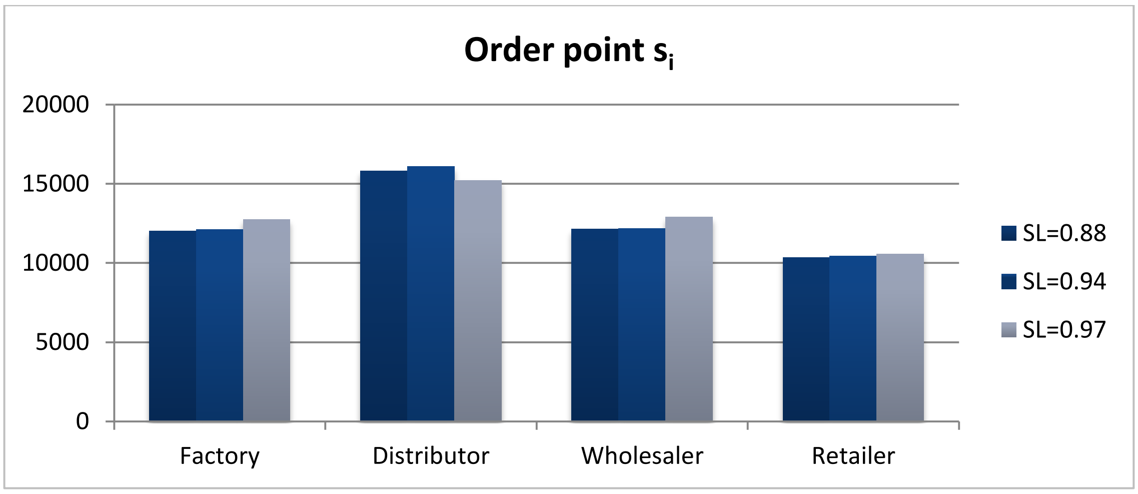


Figure 7.15: Order point s_i for the service levels: 88%, 94% and 97%

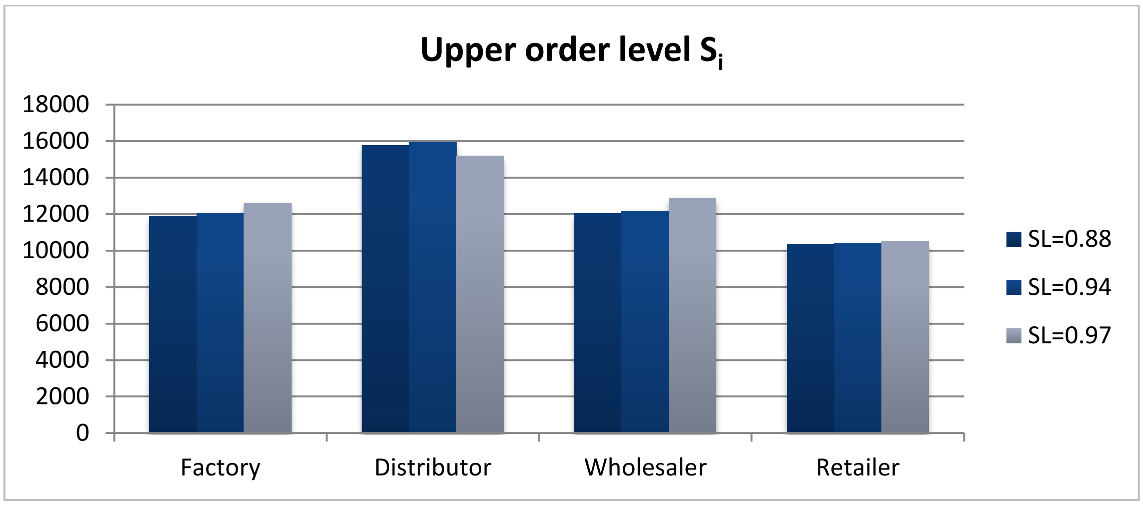


Figure 7.16: Upper level S_i for the service levels: 88%, 94% and 97%

The on-hand stock (Figure 7.17) shows that the solution for the 97% service level is greater than for the other cases. That can be caused for the high service level, as the retailer has to accomplish a 98% and has to have enough stock to satisfy on time the demand.

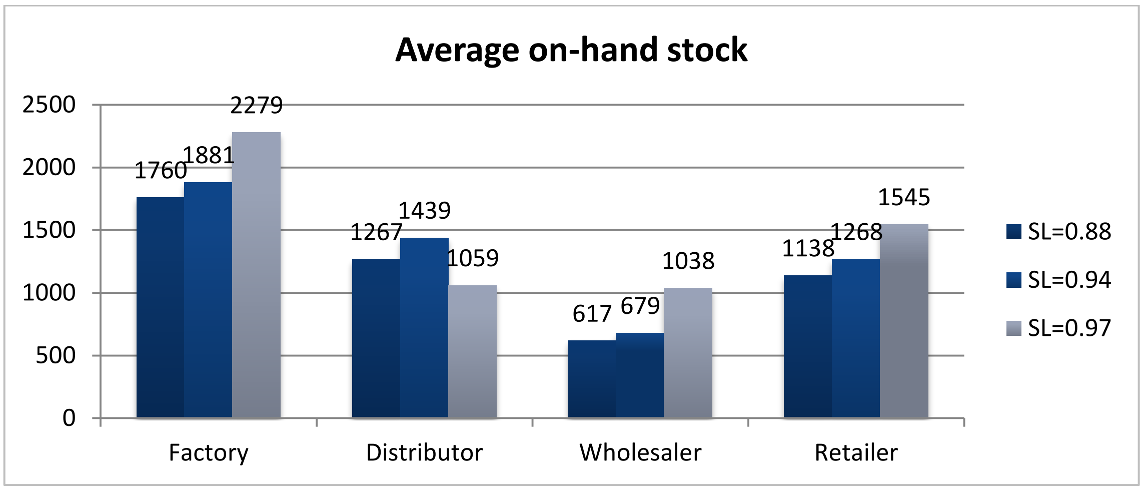


Figure 7.17: Average on-hand stock of the simulation using the solutions found for SL=88%, 94% and 97%

Ten simulations were run to enclose the daily cost for the different service levels as in the previous chapter with the (s,Q) case. The confidence interval for the mean daily cost are computed and shown on the table (Table 7.6). The intervals do not overlap as in the previous case.

Table 7.6: 95% confidence interval for the mean of the daily cost II

	95% Confidence interval for the mean
$\boldsymbol{Service\ level} = \mathbf{88\%}$	$\{3112.59, 3154.14\}$
$\boldsymbol{Service\ level} = \mathbf{94\%}$	$\{3195.22, 3243.36\}$
$\boldsymbol{Service\ level} = \mathbf{97\%}$	$\{3401.98, 3451.87\}$

In the following graph (Figure 7.18) it is shown the values in a boxplot of the 10 runs for each solution of service level. In the 2Q box are shown the second quartile and in the 3Q box the third quartile. The red cross is the mean, and it can be seen that the mean does not follow a straight line, so as in the previous case the daily cost is not linearly proportional to the service level.

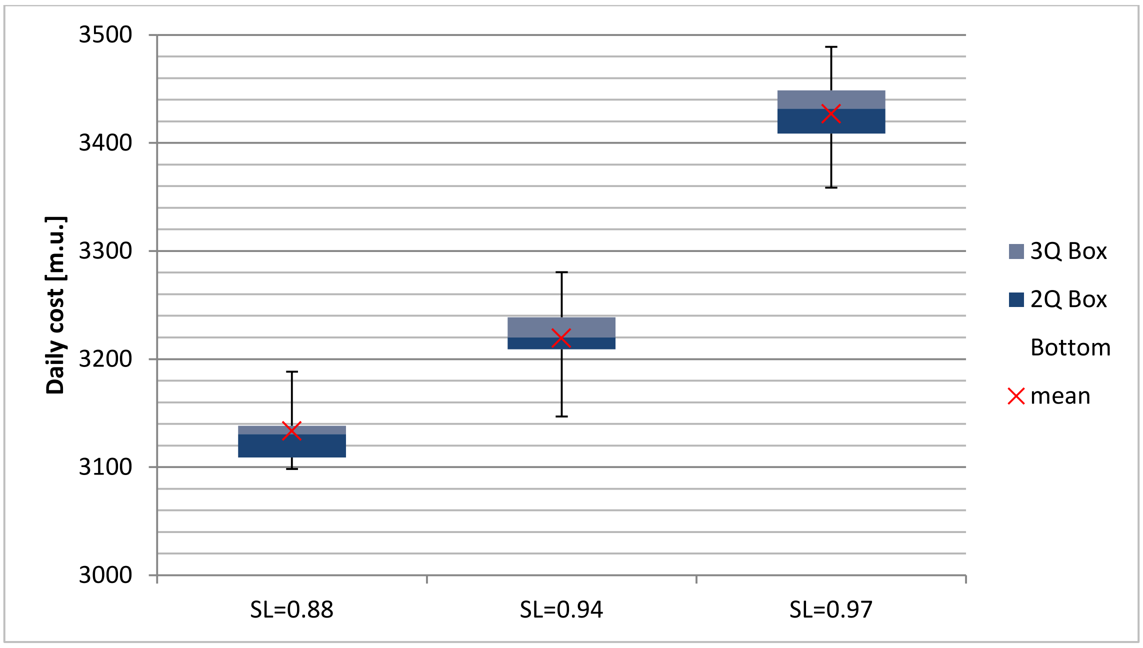


Figure 7.18: Boxplot of the runs of the solutions for service levels = 88%, 94% and 97%

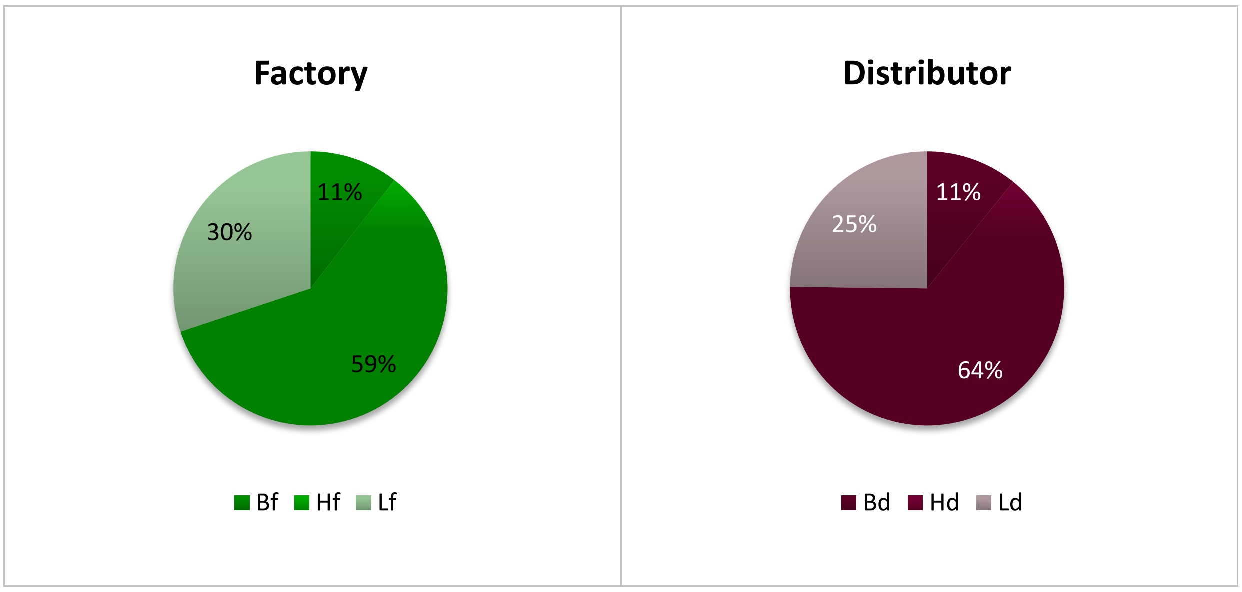
8.Discussion

(s,Q) case

In the (s,Q) case the echelon that has the more cost is the retailer, and this cost is mostly
8.1 caused by the holding cost of the on-hand inventory. In the following figure (Figure 8.1) are shown the portions of each cost over the total cost of the echelon. B_i is the backorder cost at echelon i, $i = r, w, d, f$. H_i is the holding cost at echelon i. L_i is the launching cost at echelon i, this represents the cost that incurs when an order is placed, it does not depend on the quantity ordered, only on the frequency.

The retailer only has a 2% of cost caused by the backorder cost, this happens because the retailer has to face a 95% of service level, so not much backorders are allowed.

The wholesaler also has most of the cost caused by the holding cost, but almost a quarter is caused because of the launching cost. The backorder cost is a 12% of the total cost; this is because the wholesaler does not have any service level requirement. The distributor has also the same distribution as the wholesaler. But the factory has even less holding cost portion than the others.



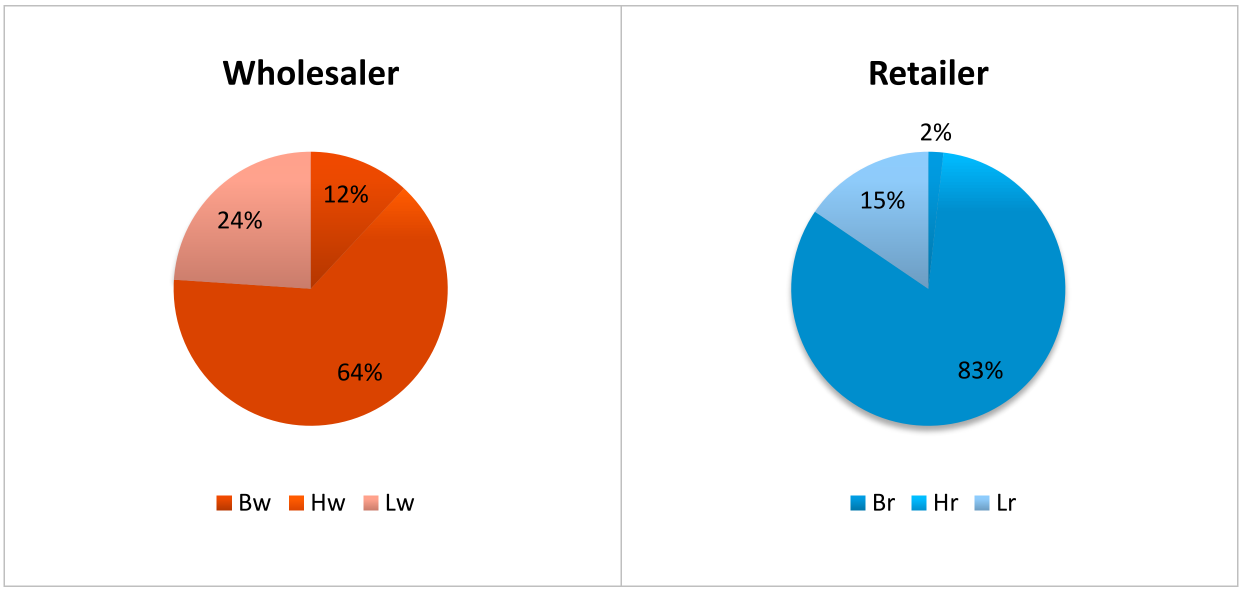


Figure 8.1: Distribution of the cost of each echelon

The following figure shows the aggregated costs of the echelons, so each colour represents a different echelon, and each tone of the colour, the different costs (Figure 8.2).

The retailer causes the 47% of the total cost, the wholesaler the 23%, the distributor the 19% and the factory the remaining 11%.

To remind (Table 8.1), the average on-hand stock of each echelon, the holding cost, and the order quantity:

Table 8.1: Avg. on-hand stock, holding cost and order quantity

	Factory	**Distributor**	**Wholesaler**	**Retailer**
Avg. on-hand stock $[pcs]$	763	751	594	1184
$h_i\left[\frac{m.u.}{pcs\cdot day}\right]$	0.2	0.4	0.6	0.8
$Q_i[pcs]$	966	644	560	420

As a simplification it could be said that the holding cost of each echelon is the avg. on-hand stock times the holding cost times the number of days. So here it can be seen that the retailer will have the greatest holding cost as is the one that has the greatest on-hand stock and holding cost. Even though the distributor and the factory have a very close average on-hand stock, as the holding cost is different, the portion over the total cost is different too. The distributor has more stock (751pcs.) than the wholesaler (594pcs.) but as the cost is greater at the latter, the cost is greater for the wholesaler than the distributor.

The total launching cost decreases in an upstream sense, this is caused because the order quantity increases in the upstream sense which means that the frequency decreases, and that leads to less cost. The launching cost per order is the same for all echelons (25m.u./order).

With exception of the retailer, the backorder costs increases in a downstream sense, caused by the increase of the unitary cost.

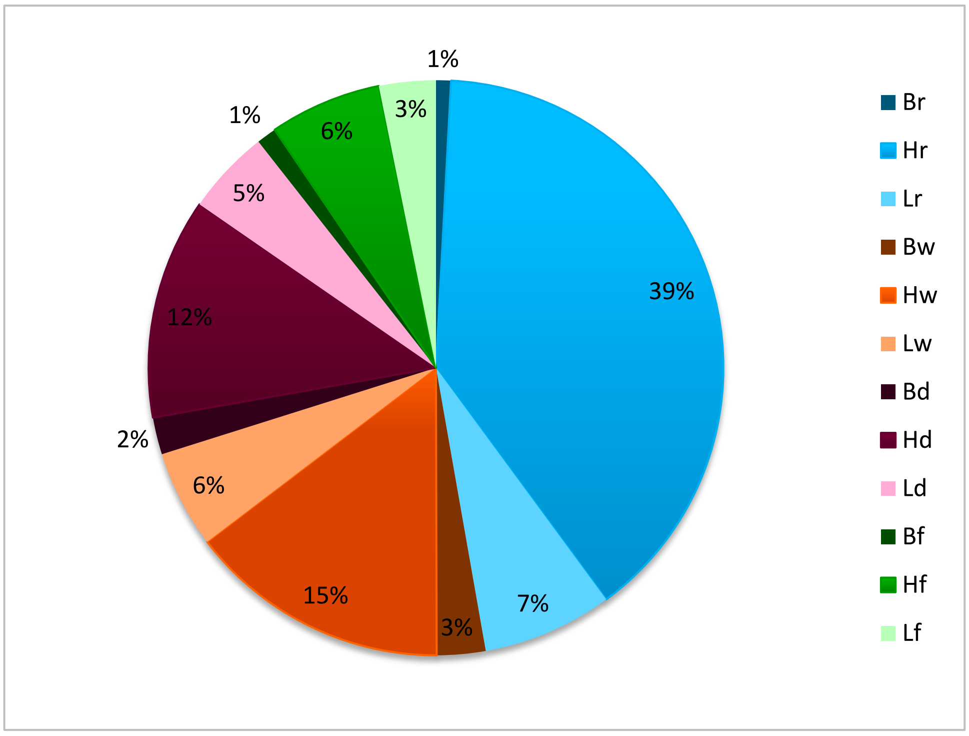
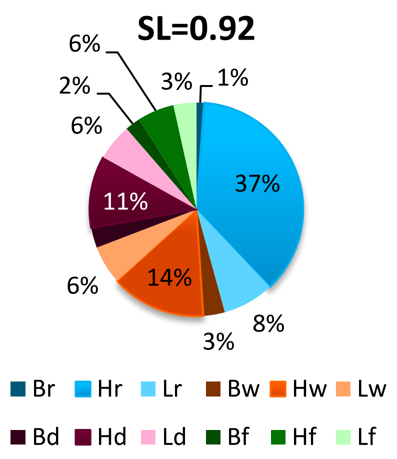


Figure 8.2: Distribution of all the costs of each echelon

8.1.1 Several service levels

It can be observed that as the service level increases, the inventory increases so the holding cost increases as well. Another main difference caused by the variation of the service level is that the backorder cost proportion decreases when the service level increases. The launching cost follows the same rule as the solution for SL=0.95 as it is still proportional to frequency of the shipments and inversely proportional to the ordering quantity.



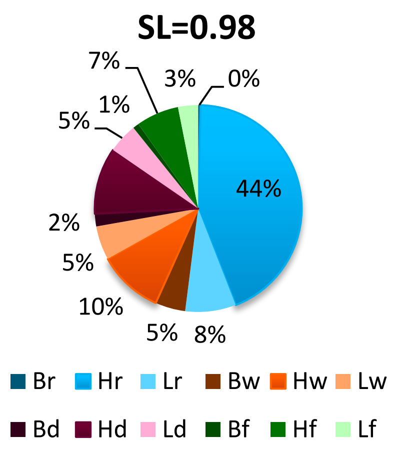


Figure 8.3: Comparison between the cost distributions of the solutions with SL=0.92 and SL=0.98

To better understand the cost, a graph (Figure 8.4) with the absolute cost is made, it can be seen that at the wholesaler, distributor and factory the difference is slight. However, at the retailer's the difference of the total cost is more important than at the other echelons. The cost increases with the service level, caused by the increase of on-hand stock at the retailer's.

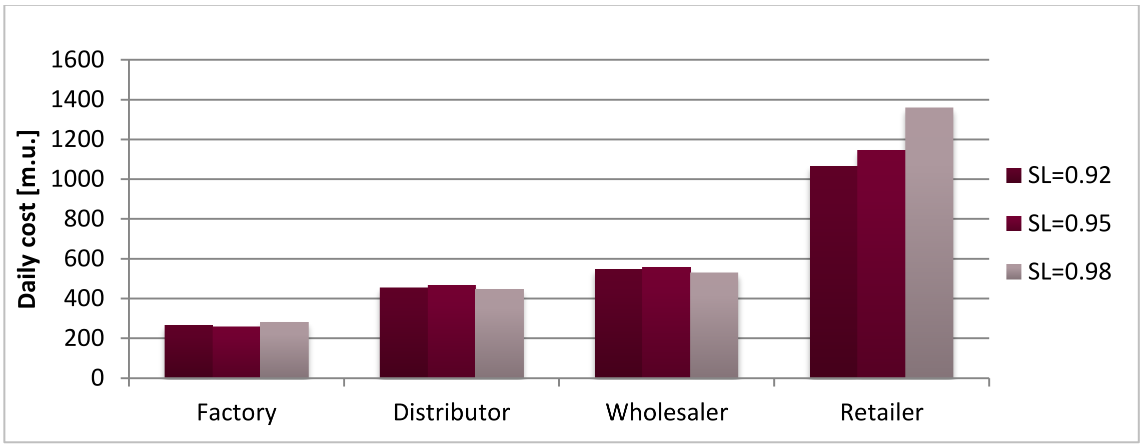


Figure 8.4: Daily cost of each echelon for each solution (SL=0.92,0.95,0.98)

8.2 (s,S) Case

The distribution of the cost for this policy is different than the (s,Q) policy (Figure 8.5). The retailer still has a really small backorder cost with respect of the other costs. The wholesaler has more than a quarter of backorder cost, but the cost is still dominated by the holding cost. The distributor has a really huge proportion of holding cost caused by the variation of orders and quantity of the downstream echelons. The factory has a big holding cost, almost three quarters of the total.

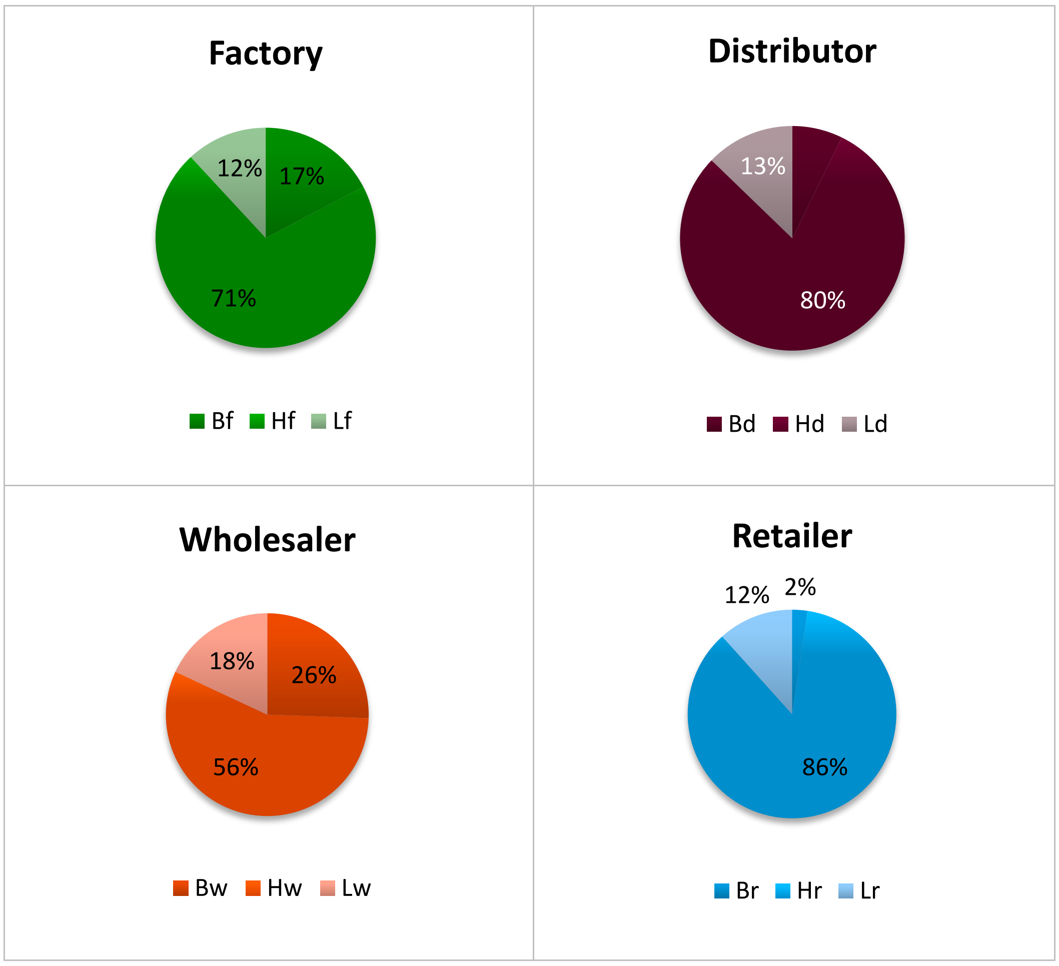


Figure 8.5: Distribution of the cost of each echelon, best solution (s,S) policy

The retailer causes the 37% of the total cost, the wholesaler a 23%, the distributor another 23% and the remaining 17% is caused by the factory (Figure 8.6).

The launching cost decreases in an upstream sense, knowing that the unitary launching cost is the same for all the echelons, that means that the frequency increases in a downstream sense.

The distributor and the factory have a big holding cost caused by the variation and the peaks of the orders coming from the downstream echelons. That is because the (s,S) model is very reactive, the echelons react too much to the peaks, so this effect increases in an upstream sense, that effect can be observed on the graph of on-hand inventory. The peaks on the inventory position cause having more on-hand inventory.

The backorders costs are small at all the echelons with exception of the wholesaler that receives the non-constant orders of the retailers almost every time step. As the orders of the retailers are not constant the wholesaler has to face more stochasticity than the retailer's, and that leads to backorders.

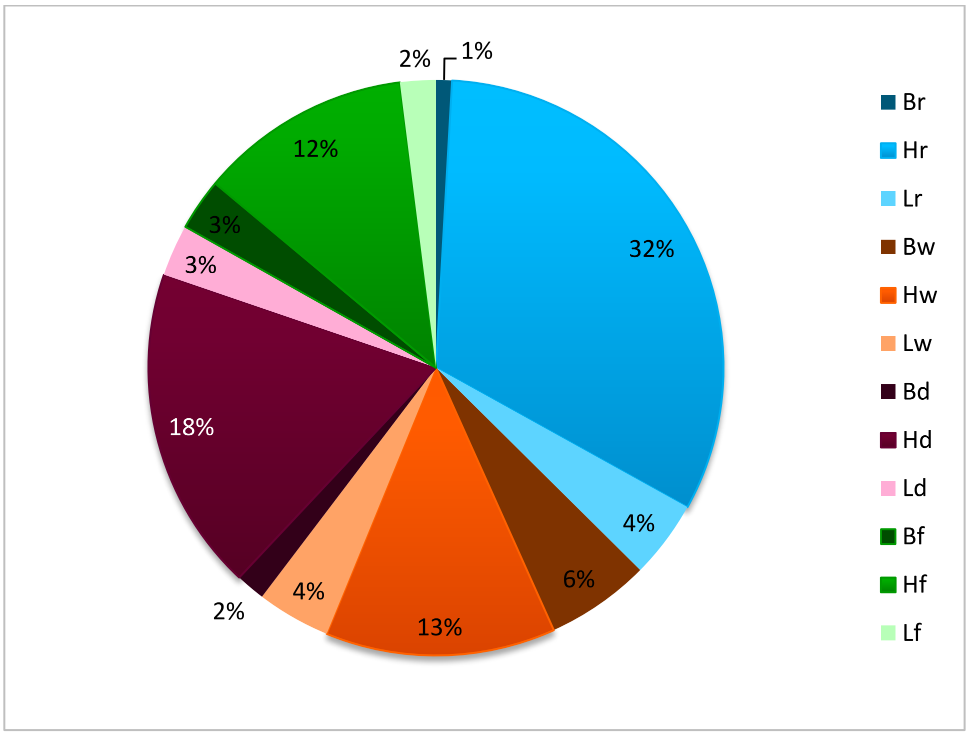


Figure 8.6: Distribution of all the costs of each echelon

8.2.1 Several service levels

For the (s,S) case the distribution of the cost for both service levels are shown below(Figure 8.7). The proportion of the holding cost at the retailers increases. There is a shift between the wholesaler and the distributor, as the distributor reduces its holding cost and the distributor increases it.

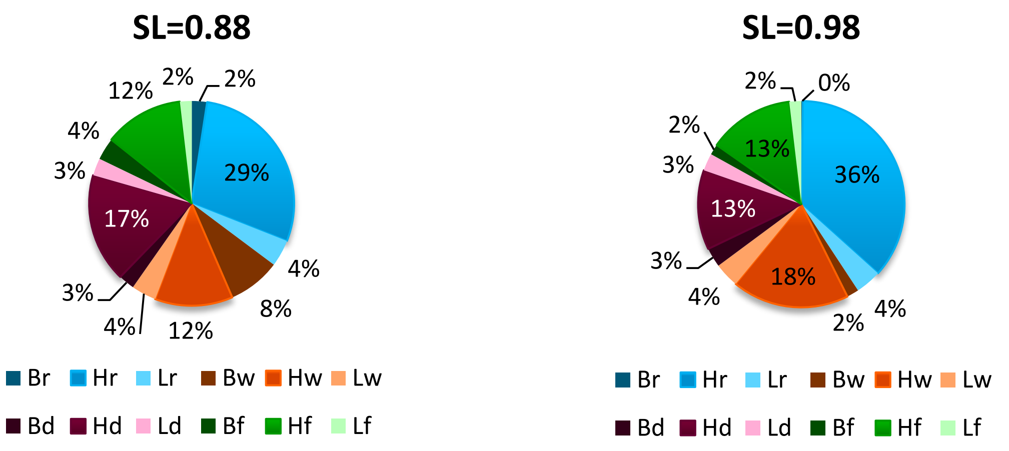


Figure 8.7: Comparison between the cost distributions of the solutions with SL=0.88 and SL=0.98

As on the other policy the increasing service level causes increasing cost, it is shown in the following figure (Figure 8.8). In this particular run the cost of the solution of SL=0.88 is higher than the solution for SL=0.94, this can happen, as seen in the previous chapter the confidence interval overlap. Almost all the cost of the SL=0.88 solution are higher than the solution of SL=0.94. But not at the retailer's, so it can be assumed that the mean would be below that point. So as in the other policy there is a clear tendency at the retailer's, where the cost increases as the service level.

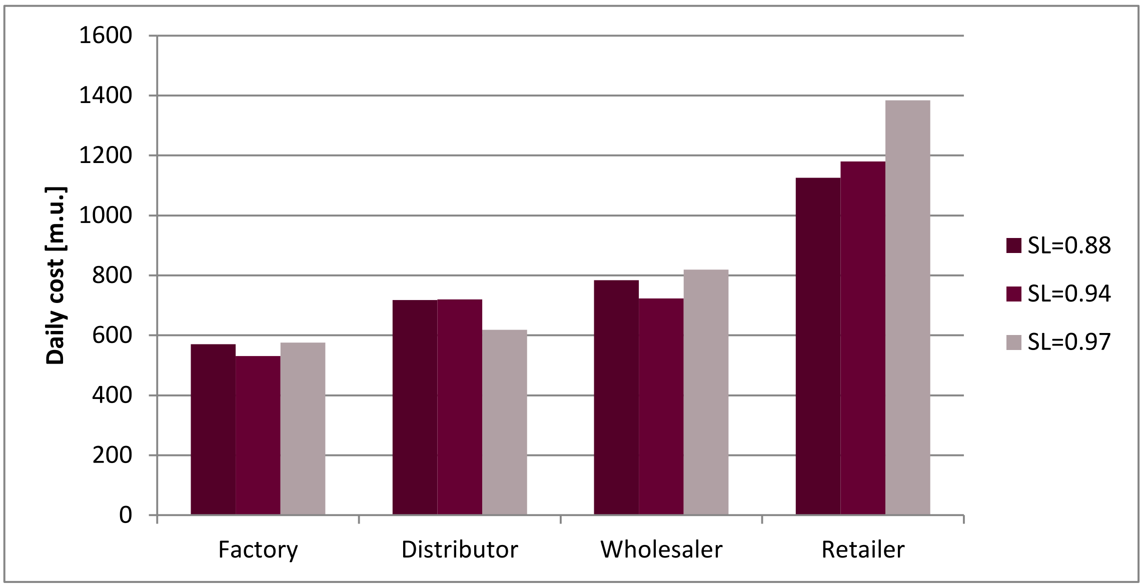


Figure 8.8: Daily cost of each echelon for each solution (SL=0.88, 0.94, 0.97)

8.3 (s,Q) and (s,S) case comparison

This chapter will provide a short summary of the cost results for the best solutions found for both policies.

One of the main differences that trigger the difference on the cost is the on-hand inventory. As it can be observed on the graph below, there is not much difference at the retailers and wholesaler, but at the distributor and factory the value for the (s,S) is the double of the (s,Q) value.

Table 8.2: Avg. on-hand stock at each echelon for the solutions of (s,Q) and (s,S) policies

On-hand stock[pcs.]	Factory	Distributor	Wholesaler	Retailer	Total
(s,Q) solution	762.7	751.1	593.5	1183.7	3291.0
(s,S) solution	1880.6	1438.6	679.4	1267.8	5266.4

On the figure below (Figure 8.9) it can be observed that the distribution of the cost is quite different. For the first case the holding cost at the retailers is almost 40% while on the other case it is 32%. In the second policy the distributor and the factory have more share than in the (s,Q) policy.

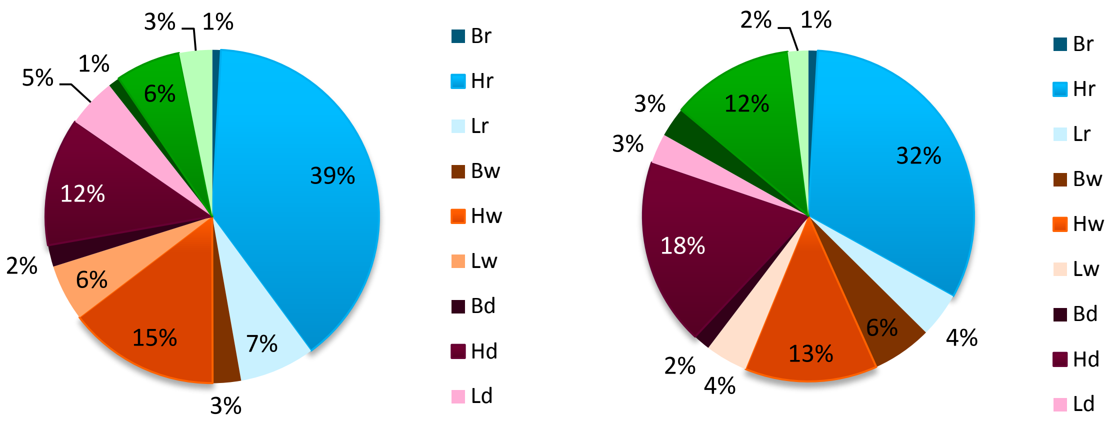


Figure 8.9: Cost distribution of the best solution of policies (s,Q) and (s,S)

The absolute cost values are also different (Table 8.3), being the cost of the (s,S) greater than the cost of the (s,Q), the confidence interval do not overlap by far.

Table 8.3: Cost and service level confidence intervals comparison

	Total cost (95% C.I.)	**Service level (95% C.I.)**
(s,Q) policy	[2387.55, 2433.03] m.u.	[0.937, 0.957]
(s,S) policy	[3195.22, 3243.36] m.u.	[0.937, 0.954]

The difference can be seen well in the following graph, where a boxplot is made and the confidence interval shown.

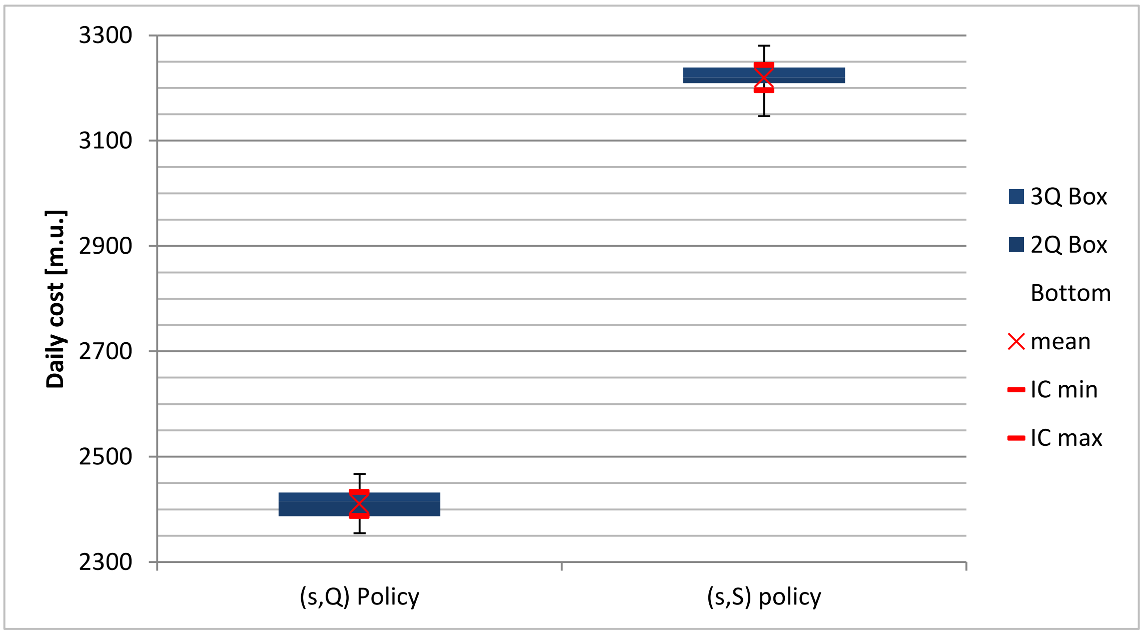


Figure 8.10: Boxplot of the daily cost of 10 runs for policies (s,Q) and (s,S)

8. Discussion

For each echelon the absolute costs are computed to show that difference at the distributor and factory is mainly caused by the holding cost (

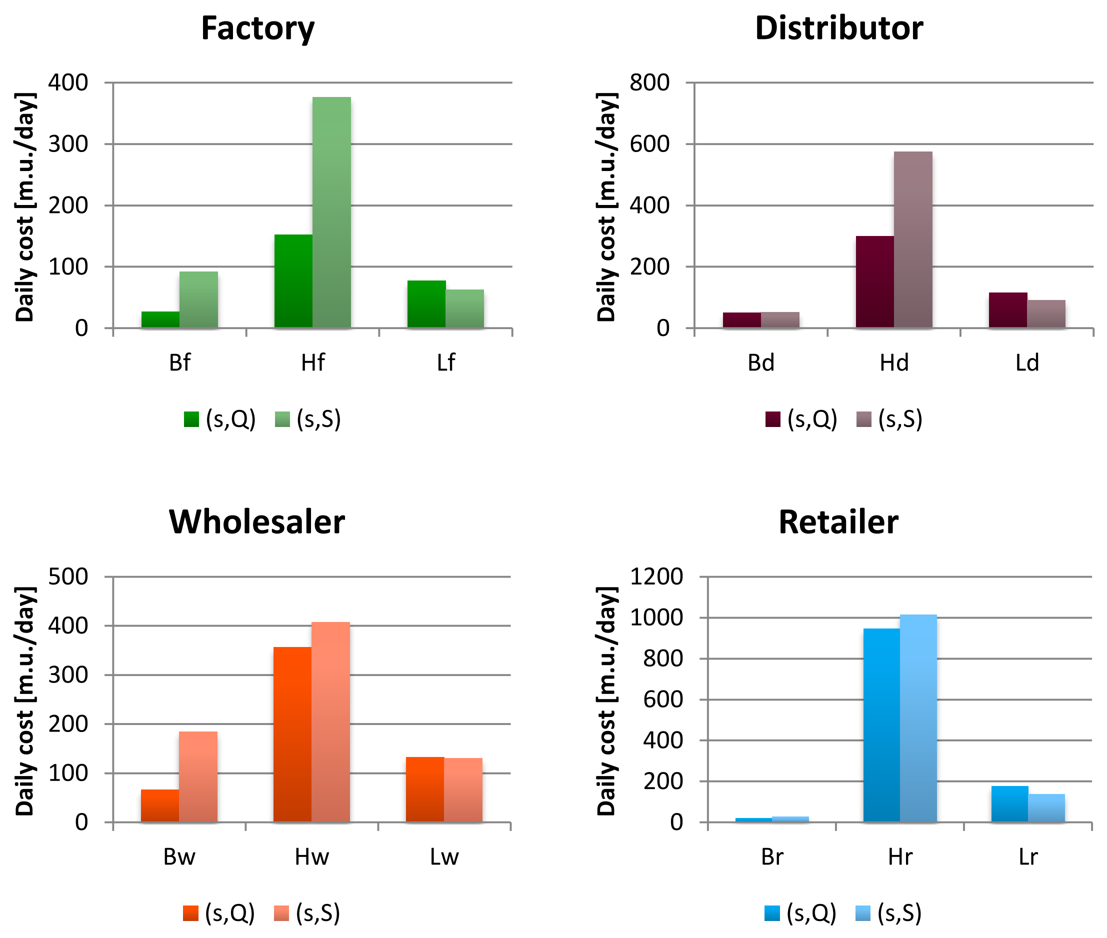
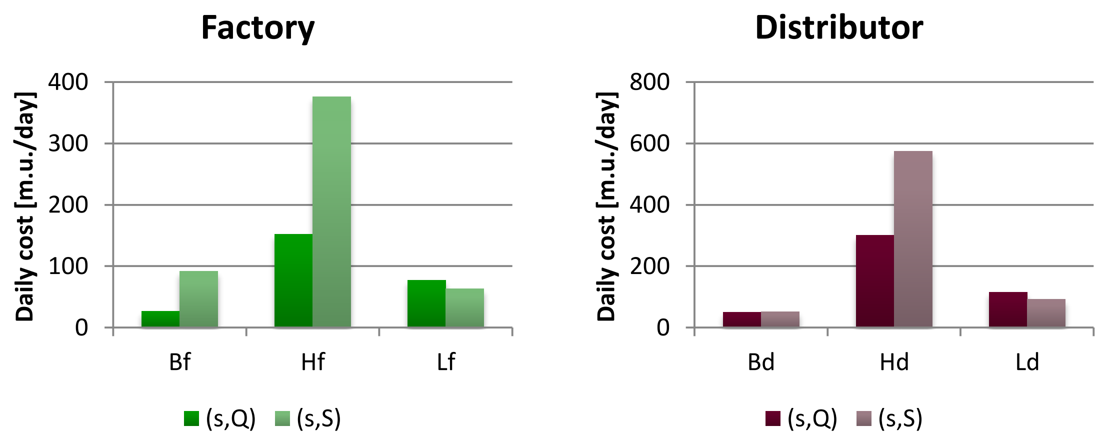


Figure 8.11). As it can be observed, the Distributor and Factory almost double the holding cost when a (s,S) policy is used instead of a (s,Q) policy. In this case, the wholesaler has more backorder cost with he (s,S) policy than in the (s,Q) policy.



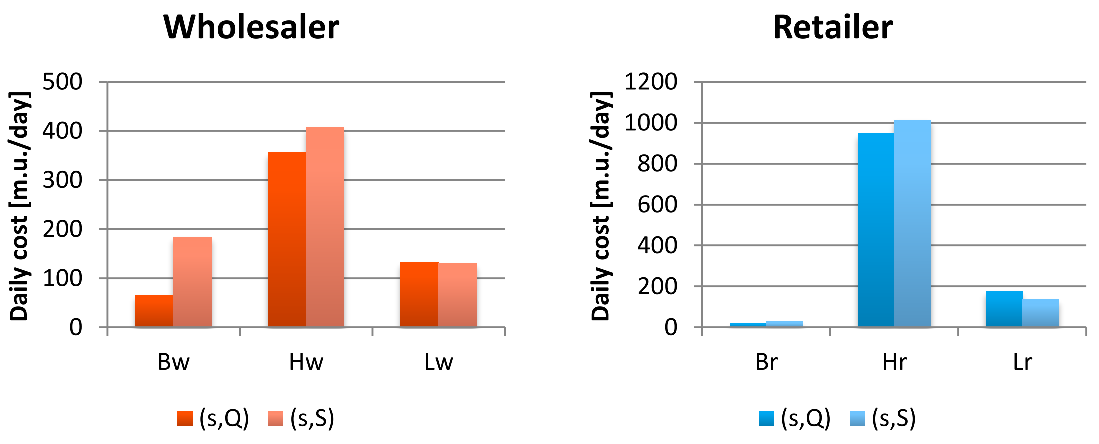


Figure 8.11: Absolute cost at each echelon for (s,Q) and (s,S) policy

This differences do not mean that the (s,S) policy is worse than the (s,Q) policy. That means that in this particular case with its unit costs, lead times and demand it is less costly to use the (s,Q) policy instead of a (s,S) policy.

9. Improvements and extensions

9.1 Improvements

The program can always be improved. In this case, some extensions were thought to be implemented in the future. Regarding the simulation, an extension would be making the parameters s, Q or S depending on the demand or orders received, that would mean that the parameters would learn as the simulation goes on. That would be applicable to a decreasing or increasing demand, for example.

An improvement regarding the Tabu search program is when a tie happens. In the Tabu search of this project, the value has to be exactly the same to consider a tie. So an improvement can be to allow some difference between the best value found and the candidate. That would mean that: being F the best value and $F_{candidate}$. A tie would occur when:

$$0.95 \times F \leq F_{candidate} \leq 1.05 \times F_{candidate} \qquad \text{(Eq. 9.1)}$$

A value $F_{candidate} < 0.95 \times F$ would be considered directly a better value and not a tie.

9.2 Extensions

The previous chapters show that there is a high difference between the two policies studied. There is a big difference as well depending on the service level restriction, that as higher the service level, the higher the cost.

In this project the global cost is analysed, but an interesting extension would be to optimize only one echelon. Only the cost of this echelon would be computed on the objective function. It could be studied what would happen if a single echelon tried to optimize its cost. Would it be really different than the result of optimizing the global cost of the chain?

An interesting extension would be to analyse how changing the cost structure can impact on the solutions and on the cost distribution. Does higher holding cost reduce the on-hand stock and increase the launching cost?

Varying the lead time may have an impact on the solution and on the ordering variables, but does it imply that the cost will vary?

Other possible extension is analysing how the variation of the demand may affect the results. In this project a constant demand has been used, but it can be changed to a cyclical, seasonal or increasing/decreasing demand. Changing the demand may affect the differences of the (s,Q) and (s,S) best cost making the difference greater or smaller.

10. Bibliography and References

Alrefaei, Mahmoud H., and Ali H. Diabat. *A simulated annealing technique for multi-objective simulation optimization.* Applied Mathematics and Computation 215 (2009) 3029–3035, 2009.

Al-Rifai, Mohammad H., and Manuel D. Rossetti. “An Efficient Heuristic Optimization Algorithm for a Two-Echelon (R, Q) Inventory System.” n.d.

Axsäter, Sven. *A framework for decentralized multi-echelon inventory contorl.* Lund, Sweden: IIE Transactions (2001) 33, 91-97, 2000.

Axsäter, Sven. *A framework for decentralized multi-echelon inventory control.* Lund, Sweden: IIE Transactions (2001) 33, 91-97, n.d.

Axsäter, Sven. *A simple procedure for determining order quantities under a fill rate constraint and normally distributed lead-time demand.* Journal of Operational Research, 174(1), 480-491., 2006.

Baron, O., O. Berman, and D. Perry. *Continuous review inventory models for.* Mathematical Methods of Operations Research, 2010.

Beamon, Benita M. *Supply Chain Design and Analysis: Models and Methods.* Seattle, WA, USA: International Journal of Production Economics, Vol. 55, No. 3, pp. 281-294, 1998.

Blum, Christian, and Andrea Roli. *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison.* ACM Computing Surveys, Vol. 35, No. 3,, 2003.

Clark, Andreu J., and Herbert Scarf. *OptimalPolicies for a Multi-Echelon Inventory Problem.* Management Science, Informs, 1960.

Duan, Q., T.W. Liao, and H.Z. Yi. *A comparative study of different local search application strategies.* LA 70803, United States: Elsevier B.V., 2013.

Keskin, Burcu B., Sharif H. Melouk, and Ivan L. Meyer. *A simulation-optimization approach for integrated sourcing and inventory policies.* Tuscaloosa,AL,USA: Computers & Operations Research 37 (2010)1648–1661, 2010.

Köchel, P., and U. Nieländer. *Simulation-based optimisation of multi-echelon inventory systems.* International Journal of Production Economics, 2005, vol. 93-94, issue 1, pages 505-513 , 2005.

Köchel, Peter, and Stefanie Thiem. *Search for good policies in a single-warehouse, multi-retaile rsystem by particle swarm optimisation.* Int. J. Production Economics 133 (2011) 319–325, 2011.

Lee, Calvin B. *Multi-Echelon Inventory Optimization.* EVANT® Inc., 2003.

Lee, Hau, and Seungjin Whang. *Decentralized Multi-Echelon Supply Chains: Incentives and Information.* Stanford, California: Graduate School of Business, Stanford University, 1999.

Li, Xiaoming. *Optimal inventory policies in decentralized supply chains.* Nashville,TN 37203,USA: Int. J.ProductionEconomics, 2010.

Moinzadeh, Kamran. *A Multi-Echelon Inventory System with Information Exchange.* Seattle, WA, USA: School of Business, University of Washington, 2001.

Respen, Jean, Nicolas Zufferey, and Edoardo Amaldi. *Metaheuristics for a Job Scheduling Problem with Smoothing Costs Relevant for the Car Industry.* Paper under review, 2014.

Rosling, Kaj. *The Square-Root Algorithm for Single-Item Inventory Optimization.* Working Paper, Växjö University (Revised for Management Science), 2002.

Sadeghi, Javad, Seyed Mohsen Mousavi, Seyed Taghi Akhavan Niaki, and Saeid Sadeghi. *Optimizing a multi-vendor multi-retailer vendor managed inventory problem: Two tuned meta-heuristic algorithms.* Knowledge-Based Systems, Volume 50, September 2013, Pages 159–170, 2013.

Tempelmeier, Horst. *A multi-level inventory system with a make-to-order supplier.* Köln, Germany: International Journal of Production Research, 51:23-24, 6880-6890, 2013.

van Houtum, G.J. *Multi-Echelon Production/Inventory Systems: Optimal Policies, Heuristics, and Algorithms.* Eindhoven, Netherlands: Tutorials in Operations Research 2006, INFORMS, 2006., 2006.

Yang, W., Felix T.S. Chan, and V. Kumar. *Optimizing replenishment polices using Genetic Algorithm for single-warehouse multi-retailer system.* Expert Systems with Applications 39 (2012) 3081–3086, 2012.

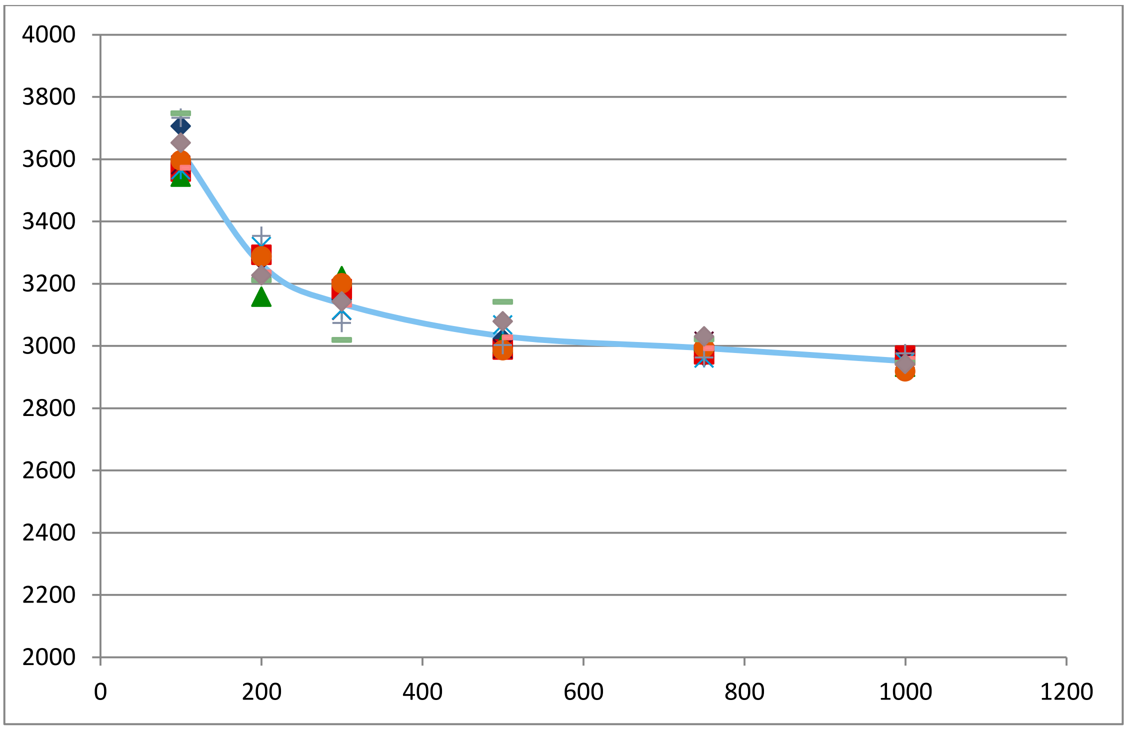
Zufferey, Nicolas. “Dynamic Tabu Search with Simulation for a Resource Allocation Problem within a Production Environment.” Proceedings of 4th International Conference on Metaheuristics and Nature Inspired Computing Sousse, Tunisia, October 27 – 31, 2012, 2012.

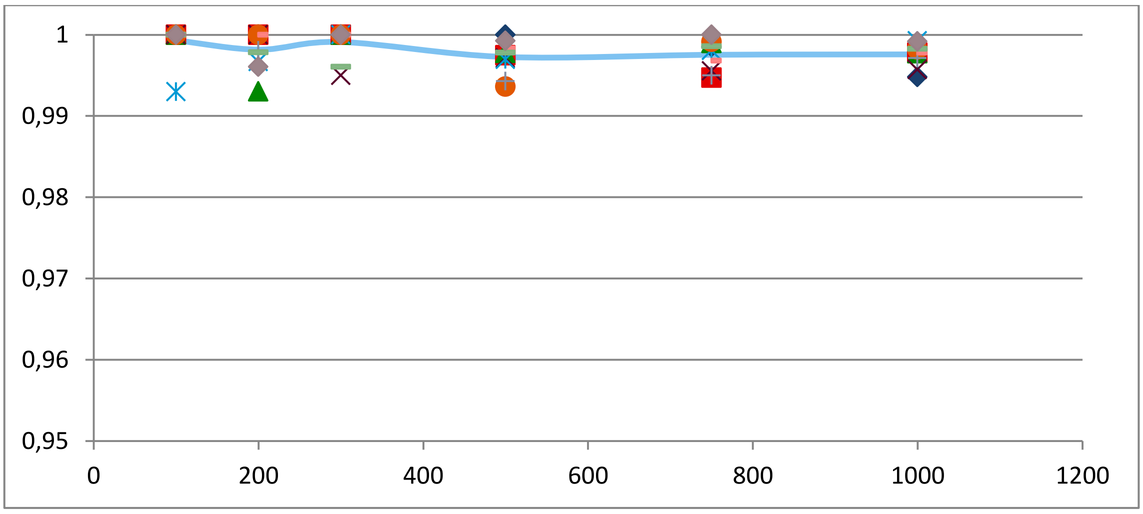
11. Appendix A: Output data

Sensitivity analysis: Simulation duration

Combination set of:

1	**s:**	{11000,	12000	15200	12000}
	Q:	{400,	550	700	800}



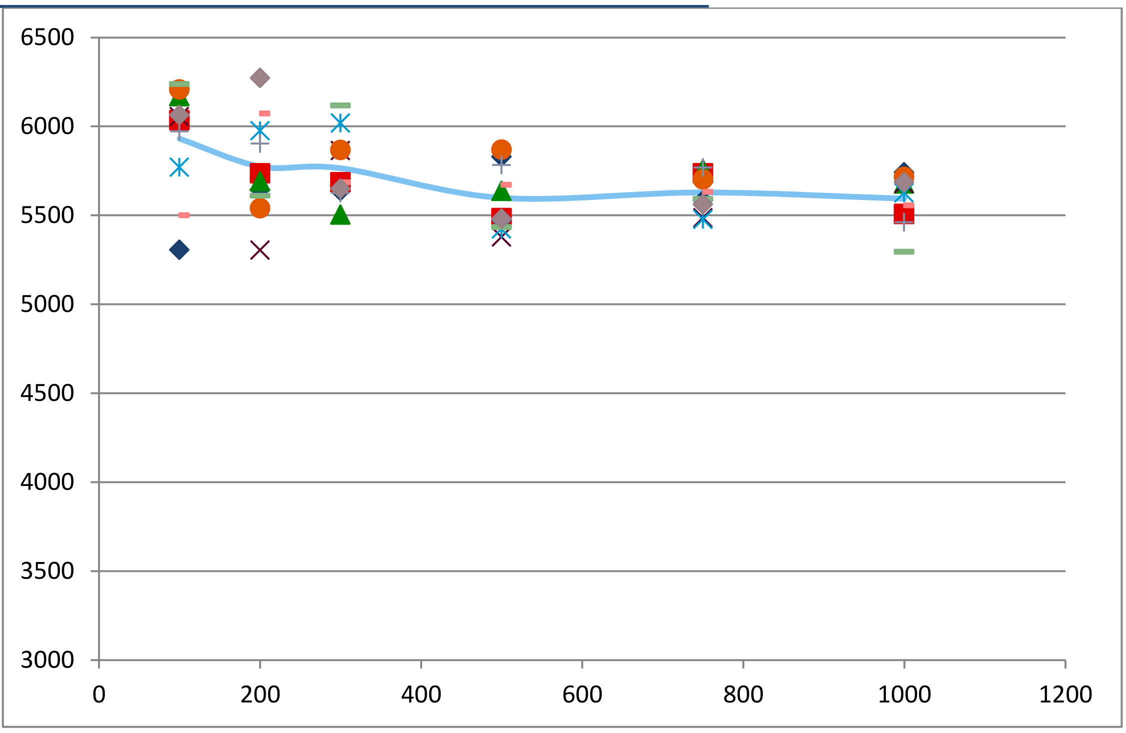


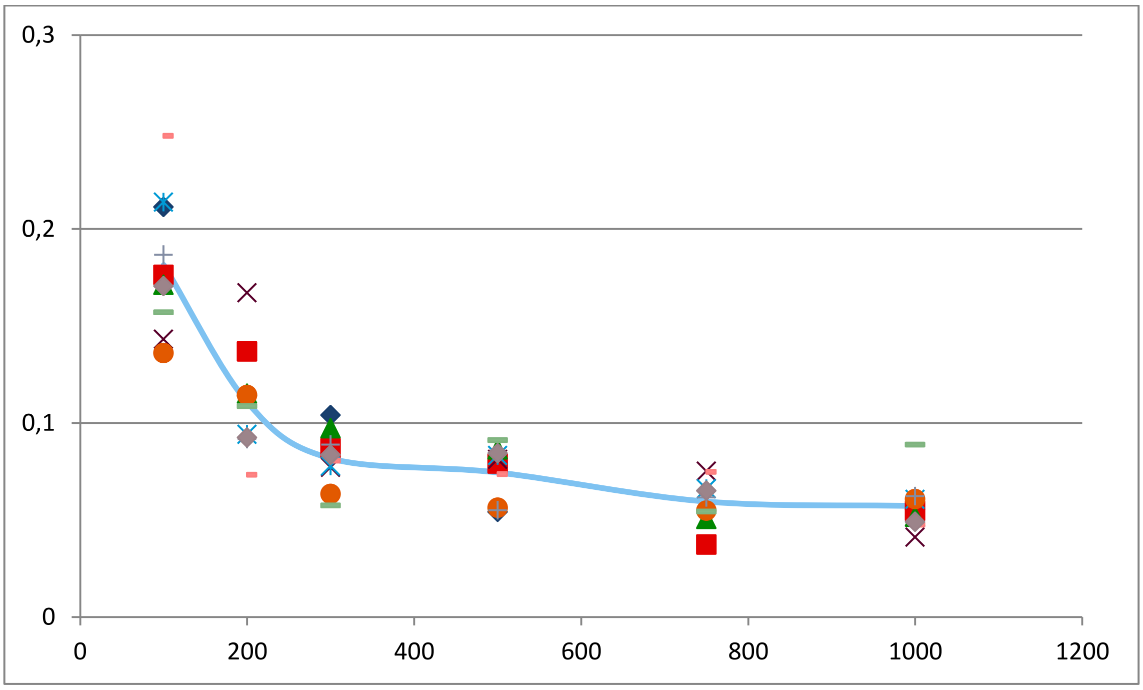
Statistical Information

11. Appendix A: Output data

	100	200	300	500	750	1000
Mean	3625,665	3261,021	3135,353	3031,532	2993,145	2951,08
Std Dev	77,20688	56,67508	60,42102	50,34162	26,2887	17,34614
CI	55,23048	40,54291	43,22259	36,01223	18,8058	12,40868
CI/mean	1,52%	1,24%	1,38%	1,19%	0,63%	0,42%

Combination set of:				
s:	{11000,	11000	13200	11000}
Q:	{400,	550	700	800}





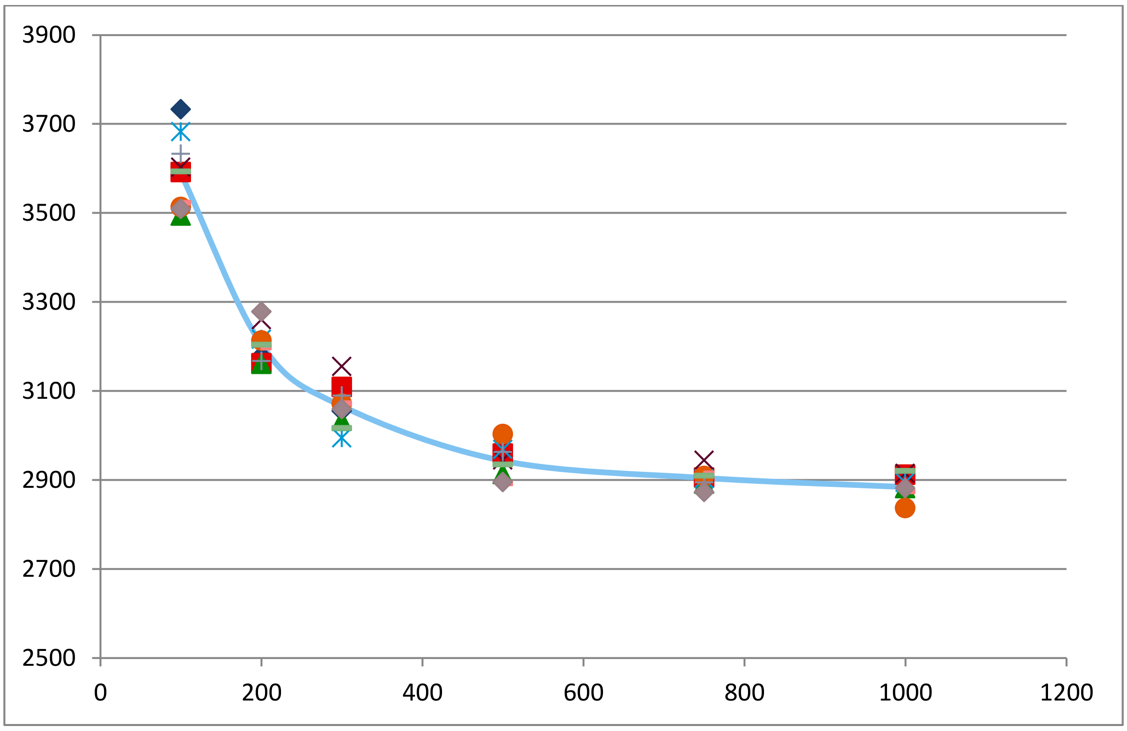
Statistical Information

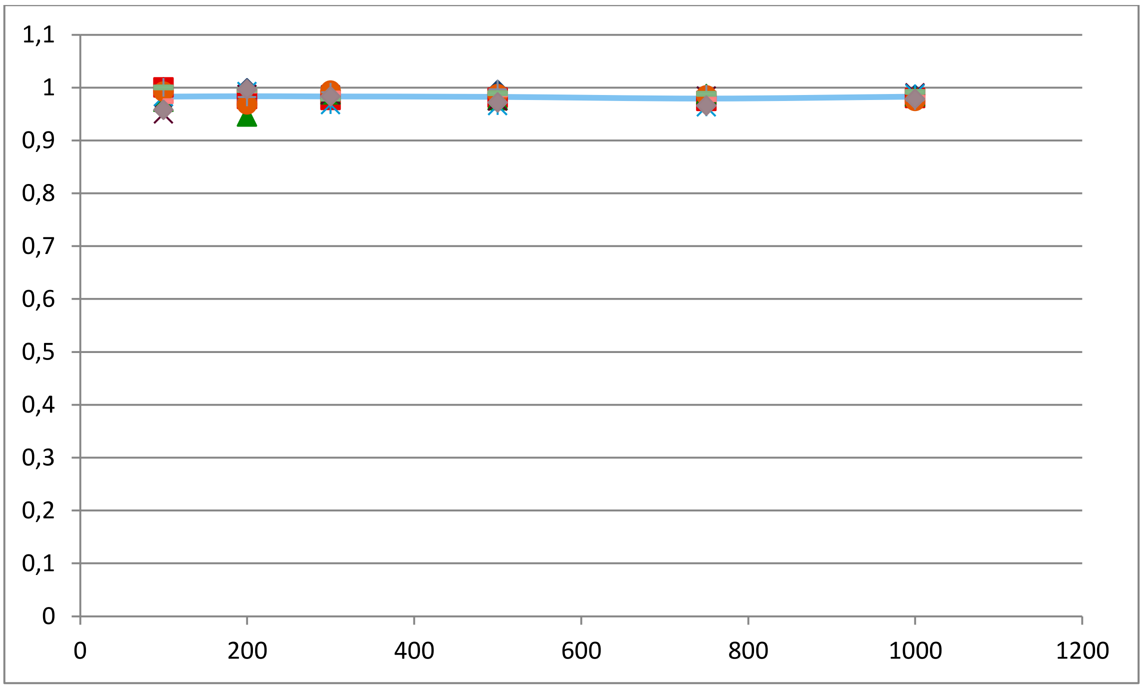
	100	200	300	500	750	1000
Mean	5930,769	5774,545	5764,78	5598,19	5628,787	5594,111
Std Dev	311,911	283,477	193,742	182,7593	107,9242	140,2526
CI	223,1277	202,7872	138,5947	130,7382	77,20433	100,3307
CI/mean	3,76%	3,51%	2,40%	2,34%	1,37%	1,79%

Combination set of:

s:	{11000,	12000	15200	11000}
Q:	{400,	400	500	800}

11. Appendix A: Output data





Statistical Information

	100	**200**	**300**	**500**	**750**	**1000**
Mean	3587,705	3204,043	3065,821	2943,074	2904,573	2884,157
Std Dev	79,67111	40,15379	46,1676	34,69438	18,22601	28,82404
CI	56,99328	28,72429	33,02631	24,81886	13,0381	20,61948
CI/mean	1,59%	0,90%	1,08%	0,84%	0,45%	0,71%

Sensitivity analysis: Tabu search parameters

For the table of the iterations, the values are in the following tables.

11.2 (s,Q)

11.2.1 $\Delta s = 100, \Delta Q = 5$

D.f.: 1 Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.95** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.9** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}	
T* (s)	F*(m.u.)	T*	F*	T*	F*
0	3203.5	0	3226.51	0	3290.79
31.159	3172.17	13.238	3152.2	19.109	3207.02
46.668	3107.66	24.54	3101.84	43.443	3158.33
65.978	3068.93	36.221	3035.42	62.841	3090.63
72.105	2999.57	51.22	2987.03	82.889	3045.71
97.125	2995.18	68.527	2970	106.794	3003.99
117.125	2969.78	85.648	2904.27	132.35	2965.65
137.72	2897.14	107.651	2877.45	161.856	2955.05
160.799	2848.64	125.039	2831.71	183.382	2878.9
181.156	2846.77	199.371	2802.4	209.762	2861.61
249.878	2828.62	217.277	2788.47	223.491	2833.21
267.385	2808.41	253.751	2784.14	237.736	2817.01
284.194	2805.36	283.184	2767.93	251.383	2792.1
301.334	2754.5	299.711	2732.94	268.481	2761.11
340.344	2699.18	317.084	2707.96	293.901	2757.33
360.779	2697.62	335.085	2685.51	310.564	2732.25
373.769	2673.77	353.222	2644.27	323.661	2707.98
482.19	2661.03	373.231	2640.43	336.575	2687.7
503.235	2657.3	393.999	2630.46	356.349	2664.11
523.702	2645.77	411.463	2609.1	369.904	2638.55
537.041	2619.41	427.527	2597.29	420.068	2634.82
546.961	2608.74	444.616	2587.19	434.925	2622.66
560.284	2581.41	513.11	2566.14	448.491	2609.12
621.237	2557.89	534.388	2560.7	482.871	2597.57
650.976	2545.79	566.134	2556.58	565.289	2589.84
676.326	2521.56	665.792	2554.93	587.389	2559.54
1389.07	2516.1	694.656	2545.74	635.802	2550.2
1403.38	2508.72	717.397	2532.58	741.329	2543.62
1461.7	2494.69	734.938	2531.75	751.023	2525.85
1859.29	2485.82	756.74	2525.8	772.514	2493.08
1995.99	2483	797.979	2516.83	781.653	2469.34
2474.28	2470.69	865.911	2493.48	1021.02	2462.19
2931.11	2466.59	948.527	2472.97	1134.39	2461.95
3600	2466.59	966.98	2470.91	1480.55	2456.85
		2744.15	2469.58	3280.35	2444.14

		3350.35	2467.57	3600	2444.14
		3504.65	2458.83		
		3600	2458.83		

D.f.: 0.85		**D.f.: 0.8**		**D.f.: 0.75**	
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}	
T* (s)	F*(m.u.)	T*	F*	T*	F*
0	3219.94	0	3204.73	0	3181.22
49.689	3149.96	20.387	3156.92	45.617	3132.96
68.11	3113.3	44.216	3135.96	70.057	3077.65
94.26	3042.68	66.397	3056.15	92.245	3053.43
124.388	3027.39	85.29	3026.58	113.87	2971.19
142.923	2980.18	107.281	2979.71	140.757	2923.96
160.469	2899.97	128.544	2931.96	163.056	2880.31
186.094	2868.16	144.808	2896.44	180.021	2838.8
212.721	2828.63	164.103	2861.54	207.073	2821.75
241.312	2814.66	185.758	2849.75	237.42	2811.69
259.541	2762.56	208.193	2822.93	250.77	2803.93
276.657	2705.48	230.445	2809.52	272.727	2784.85
325.065	2685.57	265.557	2760.23	285.162	2743.9
339.122	2662.05	282.382	2736.34	302.569	2738.33
353.763	2655.21	297.541	2734.46	322.635	2711.24
371.188	2646.38	315.367	2708.38	339.079	2647.1
409.619	2637.95	332.118	2689.27	355.857	2640.45
422.849	2619.02	340.96	2682.73	371.839	2634.27
435.813	2610.4	350.147	2651.11	380.932	2633.46
628.769	2606.96	362.061	2643.47	402.372	2620.22
647.447	2572.61	378.163	2623.75	415.141	2590.88
723.096	2571.98	408.501	2611.86	449.63	2561.14
742.157	2554.76	451.951	2608.13	564.882	2546.91
761.242	2513.08	472.268	2585.78	630.716	2534.38
1017.22	2512.66	515.414	2555.32	648.683	2517.26
1128.19	2503.74	532.538	2555.14	1139.8	2516.39
1153.53	2485.81	555.033	2552.76	1158.3	2502.65
1737.78	2485.41	704.301	2535.18	1190.69	2482.8
1832.26	2480.91	727.871	2534.41	1352.7	2464.46
1869.06	2475.11	746.376	2509.06	2933.64	2459.56
3600	2475.11	787.278	2490.03	3491.27	2451.64
		822.867	2488.05	3600	2451.64
		1704.06	2487.9		
		2427.3	2485.14		
		2776.31	2484.11		
		3179.45	2484.09		

	3600	2484.09	

D.f.: 0.7
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T*	F*
0	3322.55
27.449	3179.74
45.831	3139.19
60.168	3049.7
86.976	3015.99
113.388	2974.4
140.53	2946.09
166.464	2883.42
202.778	2876.89
264.181	2864.05
282.575	2855.47
307.883	2829.09
333.619	2773.78
379.747	2749.18
393.007	2714.32
414.814	2699.37
430.742	2697.17
443.902	2673.22
456.624	2650.21
474.242	2594.82
521.451	2583.73
534.574	2570.17
551.983	2553.36
569.884	2541.33
603.785	2517.11
902.564	2513.22
1094.45	2502.15
1514.72	2479.21
2001.27	2468.25
2053.84	2461.25
2076.95	2459.14
2219.78	2454.36
2553.07	2449.63
3269.48	2442.02
3462.2	2435.01
3600	2435.01

11. Appendix A: Output data

$$\boldsymbol{\Delta s = 500, \Delta Q = 25}$$

D.f.: 1 Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.95** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.9** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}	
T* (s)	F*(m.u.)	T*	F*	T*	F*
0	3225.46	0	3225.71	0	3192.12
8.402	2994.3	11.322	2961.3	9.762	3061.02
20.434	2805.09	25.504	2786.61	26.588	2765.09
33.55	2688.04	41.575	2677.76	41.655	2687.6
48.541	2586.02	58.245	2593.45	56.059	2579.75
75.759	2548.67	78.873	2536.77	72.197	2570.96
100.59	2541.92	151.964	2530.64	92.677	2554.95
121.136	2506.08	243.173	2521.88	129.344	2552.65
866.842	2501.5	605.698	2511.32	145.478	2536.6
946.213	2497.92	3600	2511.32	344.877	2522.08
3600	2497.92			3600	2522.08

D.f.: 0.85 Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.8** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}		**D.f.: 0.75** Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}	
T* (s)	F*(m.u.)	T*	F*	T*	F*
0	3288.4	0	3242.45	0	3231.51
11.252	2966.59	16.526	3041.52	19.966	2967.73
24.252	2781.09	34.663	2781.01	44.636	2762.35
43.025	2665.22	54.595	2660.17	65.172	2692.44
63.001	2649.19	75.055	2590.94	85.832	2586.16
92.081	2605.98	104.482	2574.41	122.764	2557.48
121.538	2599.45	125.112	2544.64	173.249	2522.33
167.379	2597.23	145.646	2543.26	533.19	2505.06
188.044	2555.87	170.714	2536.44	706.38	2493.65
213.255	2507.42	258.135	2523.81	3600	2493.65
405.734	2495.83	274.485	2510.61		
1762.59	2491.59	299.968	2502.3		
1904.99	2482.76	503.399	2476.05		
3600	2482.76	3600	2476.05		

11. Appendix A: Output data

D.f.: 0.7
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T*	F*
0	3282.55
20.238	2978.13
36.178	2792.99
56.634	2734.14
81.644	2635
118.376	2631.46
138.927	2550.21
346.996	2542.11
379.4	2530.43
490.551	2530.04
684.351	2520.13
3600	2520.13

$$\Delta s = 1000, \qquad \Delta Q = 50$$

D.f.: 1
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3181.32
8.64	2888.32
17.405	2678.84
32.386	2626.54
59.699	2624.62
101.511	2578.95
114.837	2567.18
138.131	2562.15
176.276	2539.19
3600	2539.19

D.f.: 0.95
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3274.07
8.688	2861.44
19.24	2704.57
37.375	2668.45
51.812	2661.94
60.873	2608.86
82.267	2601.83
105.57	2584.42
245.903	2546.14
3600	2546.14

D.f.: 0.9
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3248.45
9.865	2853.32
27.581	2688.33
46.138	2677.1
64.411	2617.79
116.358	2611.33
129.286	2609.11
178.649	2606.74
193.171	2601.13
285.985	2569.92
3210.42	2551.15
3233.58	2544.8
3323.13	2540.81
3600	2540.81

11. Appendix A: Output data

D.f.: 0.85
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3300.09
14.213	2848.22
37.971	2820.47
59.694	2787.72
78.167	2598.21
339.769	2585.8
688.893	2573.77
748.324	2554.05
3600	2554.05

D.f.: 0.8
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3258.73
17.067	2860.73
36.16	2692.01
54.697	2666.93
71.242	2603.5
172.21	2583.05
194.781	2571.09
209.191	2569.92
459.081	2565.51
2272.06	2548.17
2868.67	2546.71
3600	2546.71

D.f.: 0.75
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3214.53
14.261	2884.1
33.02	2671.71
51.134	2658.8
64.709	2652.71
79.802	2644.71
98.143	2634.91
125.843	2588.56
135.14	2582.55
228.53	2553.77
279.972	2529.5
3600	2529.5

D.f.: 0.7
Initial Solution: {10000, 13000, 16000, 13000, 433, 500, 612, 866}

T* (s)	F*(m.u.)
0	3206,32
19,809	2895,86
34,172	2694,42
56,515	2669,22
74,949	2648,37
107,783	2643,84
124,677	2642,72
133,757	2622,32
147,863	2580,12
170,329	2565,35
216,528	2559,83
265,346	2540,61
3600	2540,61

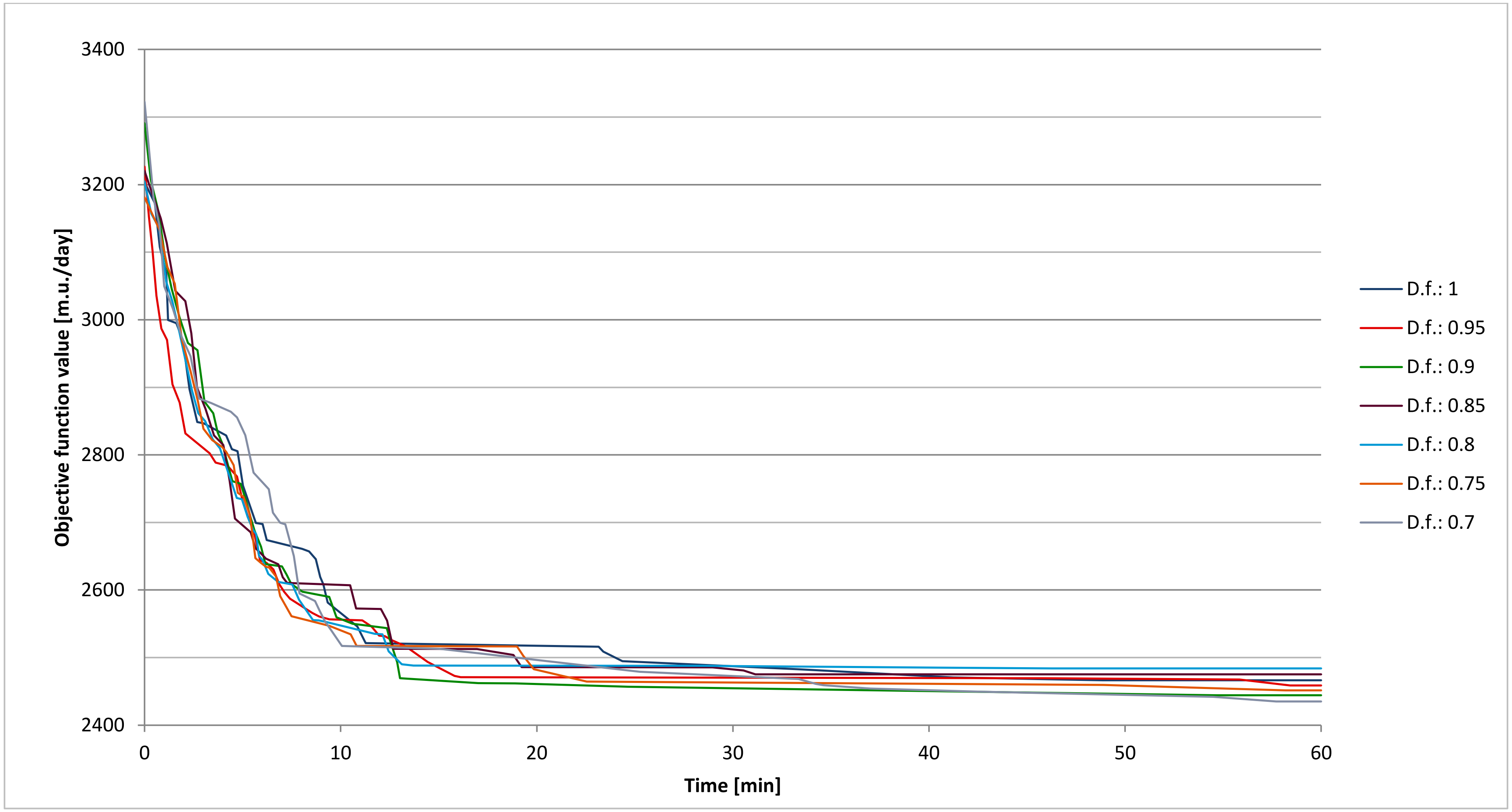


Figure 11.1: Iteration graph of the combinations of D.f. with Δs=100, ΔQ=5

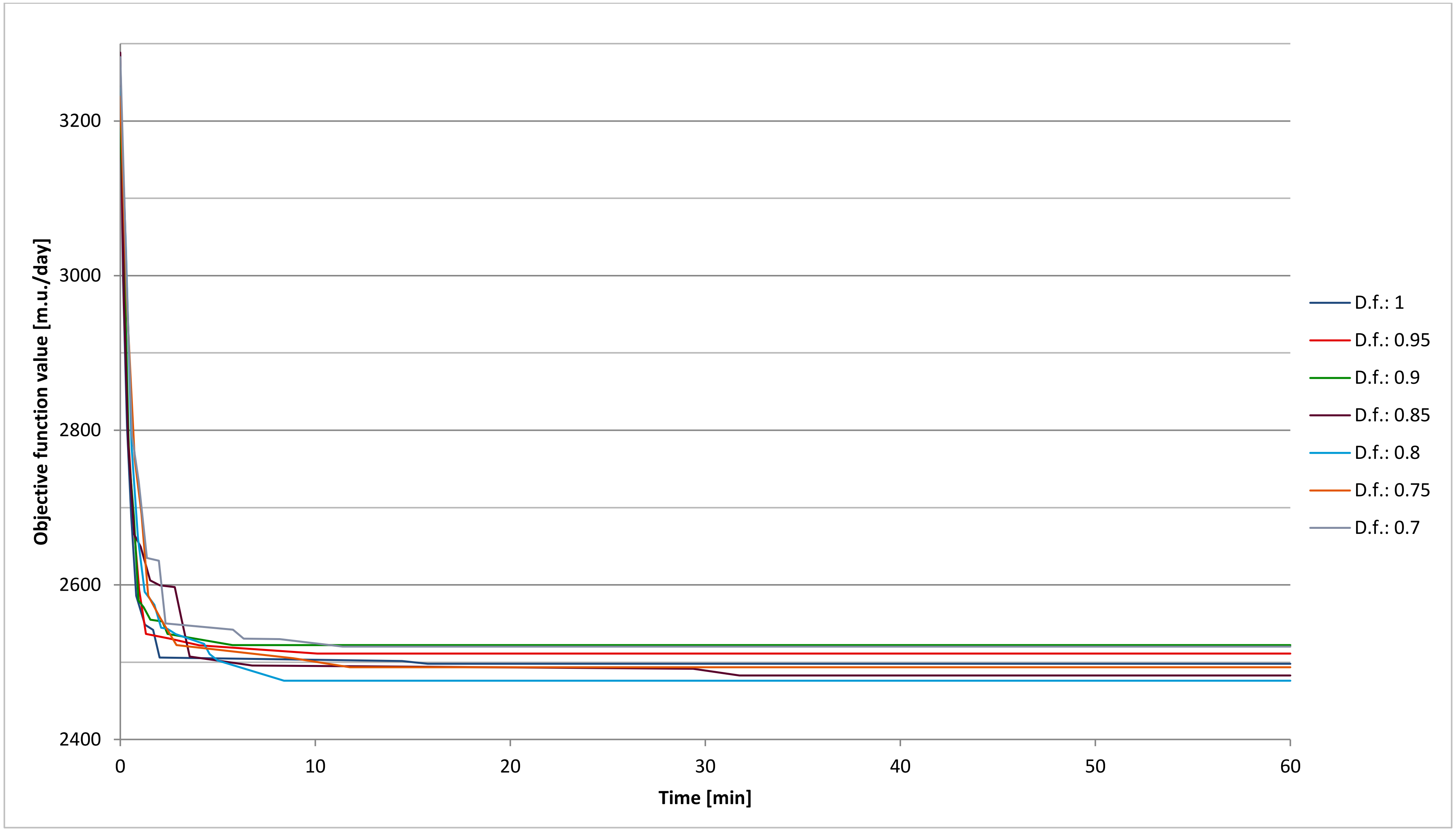


Figure 11.2: Iteration graph of the combinations of D.f. with Δs=500, ΔQ=25

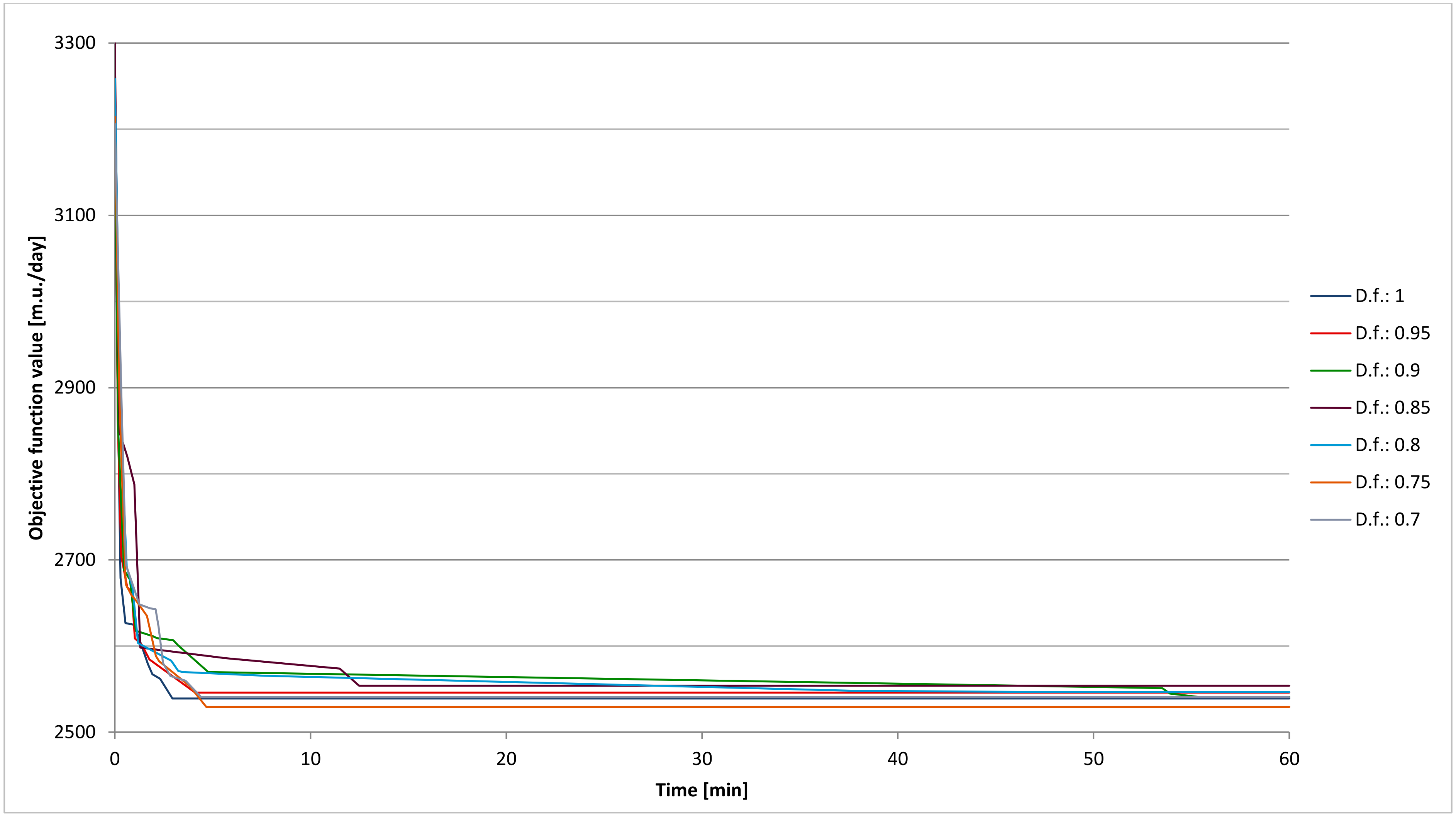


Figure 11.3: Iteration graph of the combinations of D.f. with Δs=1000, ΔQ=50

11.2.2 (s,S)

$$\Delta s = 100, \qquad \Delta S = 100$$

D.f.: 1		D.f.: 0.95		D.f.: 0.9	
T* (s)	F*(m.u.)	T* (s)	F*(m.u.)	T* (s)	F*(m.u.)
11.463	4170.5	12.23	4179.74	13.391	4158.95
22.552	4165.24	22.393	4105.23	31.76	4152.44
37.151	4151.17	56.274	4051.97	44.246	4140.76
56.328	4121.96	70.124	4051.27	63.099	4103.4
100.294	4119.14	76.829	3947.27	81.826	4100.16
120.518	4100.26	107.607	3937.64	95.428	4067.52
134.086	4080.71	114.118	3914.82	121.491	4056.02
145.121	4069.27	120.538	3889.69	137.876	4014.08
156.619	4026.14	130.442	3870.8	152.253	4008.19
167.421	4014.15	136.913	3824.68	164.004	4006.89
176.533	4012.72	150.751	3822.94	176.02	3962.83
191.861	3980.66	162.3	3776.83	191.372	3953.5
206.54	3974.62	202.504	3737.76	202.025	3914.03
231.721	3971.99	234.131	3720.73	259.839	3851.74
243.112	3952.17	250.377	3706.79	274.601	3848.42
262.683	3923.61	294.563	3677.83	285.572	3819.31
279.576	3843.08	304.48	3648.89	300.949	3800.92
310.944	3812.98	314.125	3641.44	307.587	3794.98
320.829	3805.49	325.59	3616.27	314.079	3705.58
334.313	3752.68	335.312	3608.78	363.924	3688.34
343.356	3749.94	349.527	3593.44	375.444	3682.07
350.407	3720.88	356.171	3592.36	390.834	3657.68
357.021	3715.07	371.945	3583.23	418.529	3628.06
366.248	3635.19	383.198	3553.08	433.513	3605.36
401.985	3597.56	398.721	3549.68	446.94	3601.83
413.152	3587.79	411.316	3542.08	481.099	3595.67
422.518	3546.77	427.119	3525.64	496.789	3558.58
455.134	3524.5	489.006	3510.15	523.643	3522.87
463.653	3501.02	515.906	3497.55	538.98	3480.06
477.715	3489.58	526.437	3472.62	587.719	3422.02
484.895	3484.26	553.107	3460.54	604.737	3381.07
517.637	3439.39	1538.14	3460.35	649.957	3374.02
528.086	3423.48	1590.47	3446.58	713.756	3371.97
543.276	3323.93	2166.4	3439.63	824.32	3370.73
652.457	3315.01	2380.35	3430.92	848.592	3367.12
711.039	3302.96	2401.07	3419.54	935.186	3366.27
759.36	3291.57	2428.35	3413.62	1050.07	3337.42
850.724	3278.6	2506.18	3398.91	1084.67	3324.35
1238.8	3278.48	3241.72	3397.27	1209.31	3323.49
1316.66	3271.22	3387.26	3383.61	2117.8	3312.93
1781.69	3257.36	3604.96	3378.33	2169.33	3312.34

1835.67	3254.91	4945	3375.06	2428.67	3304.86
1852.64	3248.72	5242.59	3362.18	2667.32	3299.79
2113.31	3248.49	5562.76	3354.66	2747.04	3270.64
2569	3237.66	5770.62	3345.78	2782.81	3265.03
3401.18	3234.39	6628.62	3323.82	2799.4	3248.34
4566.82	3229.12	6636.52	3322.86	3449.04	3247.98
7200	3229.12	6662.46	3311.98	3559.67	3247.24
		7200	3311.98	3586.71	3231.6
				5288.5	3226.83
				6175.19	3210.94
				7200	3210.94

D.f.: 0.85		**D.f.: 0.8**		**D.f.: 0.75**	
T* (s)	F*(m.u.)	T* (s)	F*(m.u.)	T* (s)	F*(m.u.)
15.292	4158.89	18.202	4161.25	18.079	4162.64
27.894	4155.96	28.967	4143.62	34.161	4138.08
43.858	4147.19	44.289	4113.18	58.502	4064.98
59.182	4124.09	79.719	4102.38	84.794	3999
70.658	4076.65	90.154	4093.84	99.041	3973.05
92.579	4047.57	104.545	4075.02	114.284	3958.01
105.758	4036.71	115.652	4039.26	129.873	3934.61
125.471	4029.34	126.219	4023.27	144.39	3917.24
131.594	3988	141.301	4009.16	155.628	3869.37
137.558	3979.85	152.297	3985.9	166.673	3867.12
147.713	3963.23	181.271	3939.34	182.972	3858.42
157.586	3927.96	192.665	3906.88	203.318	3823.07
168.463	3875.87	205.141	3903.43	303.093	3817.34
185.198	3859.65	211.612	3842.36	368.248	3795.3
201.86	3839.61	221.578	3830.28	388.453	3790.16
212.255	3820.97	265.542	3827.04	406.724	3745.6
223.024	3791.6	297.071	3805.97	450.309	3738.2
233.702	3789.19	312.094	3792.81	461.651	3712.45
243.638	3780.84	328.754	3782.19	473.268	3674.54
273.889	3772.72	343.39	3759.33	503.216	3648.24
283.358	3769.59	358.812	3753.36	523.278	3632.77
298.006	3747.83	373.195	3726.63	541.223	3623.02
308.769	3694.87	417.528	3716.31	555.826	3617.91
329.313	3690.37	432.771	3707.16	571.713	3613.08
339.58	3659.88	459.492	3677.21	587.122	3609.65
350.532	3638.99	470.866	3662.22	597.321	3566.58
372.825	3617.28	486.581	3647.75	608.124	3550.79
386.363	3591.41	497.584	3635.54	619.032	3549.7
397.727	3568.37	504.712	3588.74	644.596	3531.43
422.221	3565.74	535.074	3586.95	654.157	3530.48
435.973	3558.05	551.48	3565.45	676.339	3448.93

451.514	3543.37	574.368	3550.72	698.361	3433.22
476.318	3539.84	611.158	3530.4	713.323	3419.23
502.345	3536.97	646.819	3493.39	748.264	3396.84
517.433	3524.49	761.224	3475.52	764.762	3385.16
529.232	3514.44	799.697	3467.35	775.379	3372.21
539.457	3502.33	891.802	3467.05	801.779	3370.7
551.016	3491.39	968.173	3464.74	812.932	3364.62
589.426	3473.05	1064.33	3461.54	830.755	3351.87
601.821	3452.32	1075.44	3435.1	836.955	3347.11
636.326	3444.86	2057.48	3425.59	859.178	3345.7
683.181	3430.75	2074.15	3408.87	1033.31	3337.94
1672.28	3425.09	2136	3407.56	1054.66	3332.25
2391.36	3420.69	2171.95	3404.69	1082.09	3331.42
2468.84	3390.53	2878.21	3396.23	1337.99	3326.92
2813.79	3385.22	2967.18	3391.46	1382.38	3300.98
7200	3385.22	3736.91	3387.45	1426.16	3298.55
		3773.08	3387.41	1459.88	3297.6
		6066.07	3385.08	1732.09	3285.46
		6084.49	3365.54	1767.05	3268.58
		6350.15	3362.36	1840.71	3249.55
		6372.56	3361.72	1889.02	3242.58
		6394.84	3359.91	2091.14	3234.37
		6415.59	3353.5	2107.64	3231.23
		6422.37	3352.47	2370.11	3222.52
		6449.52	3339.48	3889.64	3218.68
		6821.48	3334.45	3954.17	3218.54
		6862.97	3329.98	4296.51	3212.78
		6875.49	3323.63	4647.76	3212.27
		6883.11	3304.29	5762.4	3210.37
		7172.53	3303.31	7200	3210.37
		7200	3303.31		

D.f.: 0.7

T* (s)	F*(m.u.)
19.041	4171.34
35.995	4171.21
52.008	4106.61
63.982	4066.87
74.202	4052.74
103.954	4044.24
119.665	4005.16
130.193	3969.69
140.825	3944.14
156.44	3930.42

11. Appendix A: Output data

168.092	3894.69
229.526	3873.4
249.939	3856.99
264.674	3835.08
276.88	3788.92
301.253	3740.61
327.588	3713.46
342.572	3706.67
373.613	3696.31
389.017	3659.79
419.866	3644.37
497.25	3640.94
518.348	3637.45
545.544	3610.49
571.544	3609.05
582.964	3603.73
604.506	3577.93
656.525	3555.55
671.795	3538.72
725.928	3536.83
755.346	3515.74
803.461	3471.81
874.204	3461.09
886.775	3451.33
1022.17	3444.94
2308.18	3444.53
2324.03	3439.02
2347.21	3434.77
2483.99	3429.69
2507.09	3427.05
2659.21	3404.79
2833.77	3401.63
5337.97	3394.29
7200	3394.29

11. Appendix A: Output data

$$\Delta s = 500, \qquad \Delta S = 500$$

D.f.: 1		D.f.: 0.95		D.f.: 0.9	
T* (s)	F*(m.u.)	T* (s)	F*(m.u.)	T* (s)	F*(m.u.)
9.859	4149.12	7.798	4178.89	7.869	4161.82
15.859	4105.31	11.734	4033.89	14.441	3990.14
24.469	4087.81	17.798	3983.6	28.169	3945.44
31.149	4008.51	22.255	3860.15	34.416	3909.25
42.174	3941.93	28.829	3825.77	38.596	3847.03
51.172	3939.51	35.602	3755.43	47.358	3826.48
55.532	3803.65	44.813	3733.96	51.462	3735.65
69.327	3760.69	49.299	3674.47	57.921	3720.02
187.335	3706.04	100.238	3667.83	106.366	3717.75
217.261	3695.22	127.223	3655.59	127.614	3698.34
259.2	3686.11	375.455	3655.34	134.028	3674.42
306.172	3681.07	806.72	3653.34	2886.58	3654.12
850.768	3676.55	1549.66	3649.84	2978.69	3650.22
865.592	3670.35	2297.54	3642.31	3068.74	3635.48
875.24	3667.05	5684.01	3638.03	4160.74	3630.01
2078.89	3664.66	6633.75	3601.4	4181.81	3624.98
2112.47	3655.59	7045.53	3580.94	4357.25	3619.39
7200	3655.59	7052.35	3567.57	4560.03	3604.89
		7200	3567.57	4765.72	3596.33
				4792.74	3594.83
				4831.1	3587.35
				4859.9	3562.17
				4881.56	3534.03
				5006.25	3521.13
				5181.46	3507.3
				5185.72	3507.18
				5312.12	3499.09
				5407.73	3496.75
				5455.32	3482.33
				7200	3482.33

11. APPENDIX A: OUTPUT DATA

D.f.: 0.85		**D.f.: 0.8**		**D.f.: 0.75**	
T* (s)	F*(m.u.)	T* (s)	F*(m.u.)	T* (s)	F*(m.u.)
11.6	4160.62	7.764	4175.17	9.786	4134.52
18.472	4004.82	11.744	4004.82	13.739	3980.06
27.34	3967.26	19.371	3998.05	19.344	3956.28
31.685	3928.79	27.136	3951.52	23.097	3867.32
41.943	3684.79	32.461	3945.36	30.714	3846.23
57.205	3637.66	36.024	3796.06	36.473	3778.85
69.806	3634.35	43.445	3769.54	41.839	3770.64
87.015	3623.29	49.079	3733.78	45.399	3693.04
93.146	3592.98	54.528	3720.14	50.853	3677.24
415.72	3590.12	119.3	3703.64	54.476	3648.69
422.18	3586.64	133.19	3694.48	1544.69	3646.78
438.53	3561.38	179.35	3679.44	1553.45	3645.45
703.04	3557.79	281.72	3652.42	1674.06	3636.12
709.8	3554.81	494.05	3646.5	1680.61	3621.75
775.48	3554.17	1516.3	3597.28	1734.73	3616.06
1709.1	3549.66	1846.9	3569	1748.08	3610.89
1985.9	3543.8	1863.5	3558.07	1769.41	3609.89
2033.2	3539.81	1877.5	3549.63	1772.98	3609.06
2041.5	3527.07	2002.2	3541.96	1851.89	3599.69
2356.9	3524.23	2010.4	3522.3	1906.95	3593.34
2563.1	3513.82	2155.6	3513.3	2005.33	3567.68
2753.4	3509.52	2387.3	3505.55	2083.63	3562.5
2991.6	3508.08	2687.5	3457.18	2109.9	3545.58
3052.7	3507.25	2703.1	3425.42	2179.84	3539.51
3123.3	3505.32	2744	3420.73	2223.14	3526.4
3309.8	3503.16	3323.4	3419.22	2352.76	3508.01
3516.1	3501.5	4240.5	3417.13	2394.38	3499.36
4494.6	3499.62	4387.9	3370.82	2549.81	3480.15
4849.1	3481.67	4396.3	3362.94	2961.44	3461.22
4857.5	3459.53	5972.4	3362.55	2991.2	3454.98
5037.9	3421.36	6247.6	3361.08	3068.14	3436.7
5153.8	3413.12	7200	3361.08	3076.98	3419.94
5409	3389.17			3247.38	3417.18
5438.1	3354.46			3294.3	3412
5455.9	3353.41			3305.06	3408.12
5562.1	3343.02			3317.92	3395.66
5628.9	3340.45			3337.15	3392.32
5702.9	3337.19			3625.56	3384.67
5729.7	3328.42			3632.38	3362.1
5777.4	3326			4024.27	3359.11
5797.1	3320.25			4136.14	3351.06
5803.6	3300.45			4176.66	3327.37
5820.5	3287.13			4184.37	3326.49
5895.7	3285.6			4201.77	3325.99

5966.6	3278.02			4242.94	3324.62
6053.7	3276.14			4254.12	3323.36
6185.9	3263.46			4353.57	3300.93
6205.8	3244.49			4438.04	3293.07
7021.4	3240.04			4494.2	3285
7028.5	3235.39			4556.52	3283.43
7046.1	3208.19			4640.32	3274.31
7200	3208.19			4668.3	3269.65
				4893.66	3269.46
				4976.25	3260.3
				5476.48	3258.92
				5628.65	3254.05
				5711.77	3249.78
				5909.28	3248.7
				5929.37	3244.36
				6009.76	3241.15
				6200.74	3240.9
				6392.27	3237.26
				6576.96	3231.75
				6603.49	3225.61
				7200	3225.61

D.f.: 0.7	
T* (s)	F*(m.u.)
9.896	4140.69
13.806	4028.73
21.352	3985.26
29.083	3951.5
34.461	3921.62
37.979	3771.78
45.38	3755.14
48.909	3718.67
56.447	3714.95
75.159	3692.74
382.699	3688.06
390.483	3679.22
404.944	3678.21
418.08	3671.84
431.002	3662.09
2253.65	3637.22
2270.4	3636.95
2278.8	3632
2588.15	3631.18
2606.3	3597.84
2727.38	3595.9

11. Appendix A: Output data

2735.52	3585.2
2814.21	3580.96
2830.79	3576.71
2874.5	3567.64
2883.51	3552.72
2896.44	3545.99
2973.81	3541.87
3004.24	3521.06
3017.48	3509.52
3579.83	3499.99
3658.72	3495.1
3839.86	3492.4
3904.47	3481.45
3914.98	3472.45
3930.24	3470.87
3967.04	3465.55
3978.53	3459.3
3994.61	3443.08
4437.93	3440.75
4463.35	3435.16
4508.96	3430.61
4599.17	3424.36
5629.59	3421.03
5672.28	3420.49
5766.47	3415.86
5889.27	3415.67
5927.5	3409.01
7200	3409.01

$$\Delta s = 1000, \qquad \Delta S = 1000$$

D.f.: 1		**D.f.: 0.95**		**D.f.: 0.9**	
T* (s)	F*(m.u.)	T* (s)	F*(m.u.)	T* (s)	F*(m.u.)
8.898	4138.16	12.715	4136.79	14.602	4132.65
17.267	3982.32	22.66	4003.95	26.344	3977.81
39.61	3744.27	33.814	3999.33	54.025	3728.92
69.678	3722.73	41.725	3712.13	129.919	3720.07
84.529	3719.96	223.483	3711.26	194.356	3707.23
144.835	3718.94	245.03	3708.09	318.988	3690.07
293.872	3718.85	412.278	3705.78	4623.13	3689.26
320.545	3718.77	907.668	3693.8	4642.94	3672.86
397.044	3702.86	1977.64	3692.89	7200	3672.86
591.065	3702.19	5636.19	3691.1		
1089.44	3698.2	6806.67	3684.42		
7200	3698.2	7200	3684.42		

11. Appendix A: Output data

D.f.: 0.85	
T* (s)	F*(m.u.)
9.056	4154.59
19.989	3862.92
31.192	3813.82
39.716	3725.77
251.79	3709.99
634.836	3707.46
891.271	3705.36
1038.01	3698.68
1677.63	3695.57
1721.68	3691.85
1824.49	3685.64
2702.21	3669.71
2770.21	3668.74
3070.29	3653.91
4778.9	3644.05
6338	3616.12
6384.54	3604.88
6605.2	3602.49
7146.28	3595.73
7178.22	3579.33
7200	3579.33

D.f.: 0.8	
T* (s)	F*(m.u.)
14.121	4160.59
25.356	3827.37
36.984	3816.77
48.343	3715.72
126.895	3711.45
631.016	3710.88
727.787	3710.62
778.982	3708.37
1059.97	3706.85
1264.3	3698.6
1680.37	3691.81
2284.92	3682.01
2296.83	3674.6
2396.58	3663.98
2857.07	3656.7
4015.96	3640.99
7200	3640.99

D.f.: 0.75	
T* (s)	F*(m.u.)
10.898	4134.81
22.917	3842.08
34.217	3838.86
42.07	3726.46
52.491	3720.17
174.579	3711.66
287.276	3705.5
407.766	3704.65
972.686	3701.66
2867.83	3676.87
2934.08	3668.82
3031.2	3663.52
3385.33	3656.32
3401.57	3631.5
3428.24	3624.16
3446.21	3615.16
3463.95	3601.61
3732.71	3572.14
3777.98	3564.87
3909.87	3551.82
4250.51	3540.28
4266.24	3531.47
4734.9	3529.81
4779.94	3516.35
4793.55	3509.71
5676.62	3508.66
6353.05	3497.62
6887.45	3487.16
7200	3487.16

D.f.: 0.7	
T* (s)	F*(m.u.)
15.589	4149.72
28.409	3980.73
51.875	3735.75
58.695	3716.68
196.345	3710.58
247.091	3708.98
479.411	3707.49
662.38	3703.01
711.696	3694.73
2113.64	3660.53
2124.48	3658.82

11. Appendix A: Output data

2274.12	3651.39
2421.22	3650.44
2934.54	3637.75
3703.64	3634.12
3752.85	3626.9
3847.85	3625.73
3865.51	3623.66
3893.52	3620.31
4161.34	3619.92
4263.46	3615.67
4689.5	3611.74
4797.65	3610.57
4983.4	3598.96
4997.91	3586.91
5086.07	3563.01
5367.73	3554.99
5407.65	3535.04
5745.3	3525.07
5776.69	3524.26
5821.77	3517.82
6573.75	3515.94
6790.36	3503.2
7069.7	3496.75
7181.02	3467.79
7200	3467.79

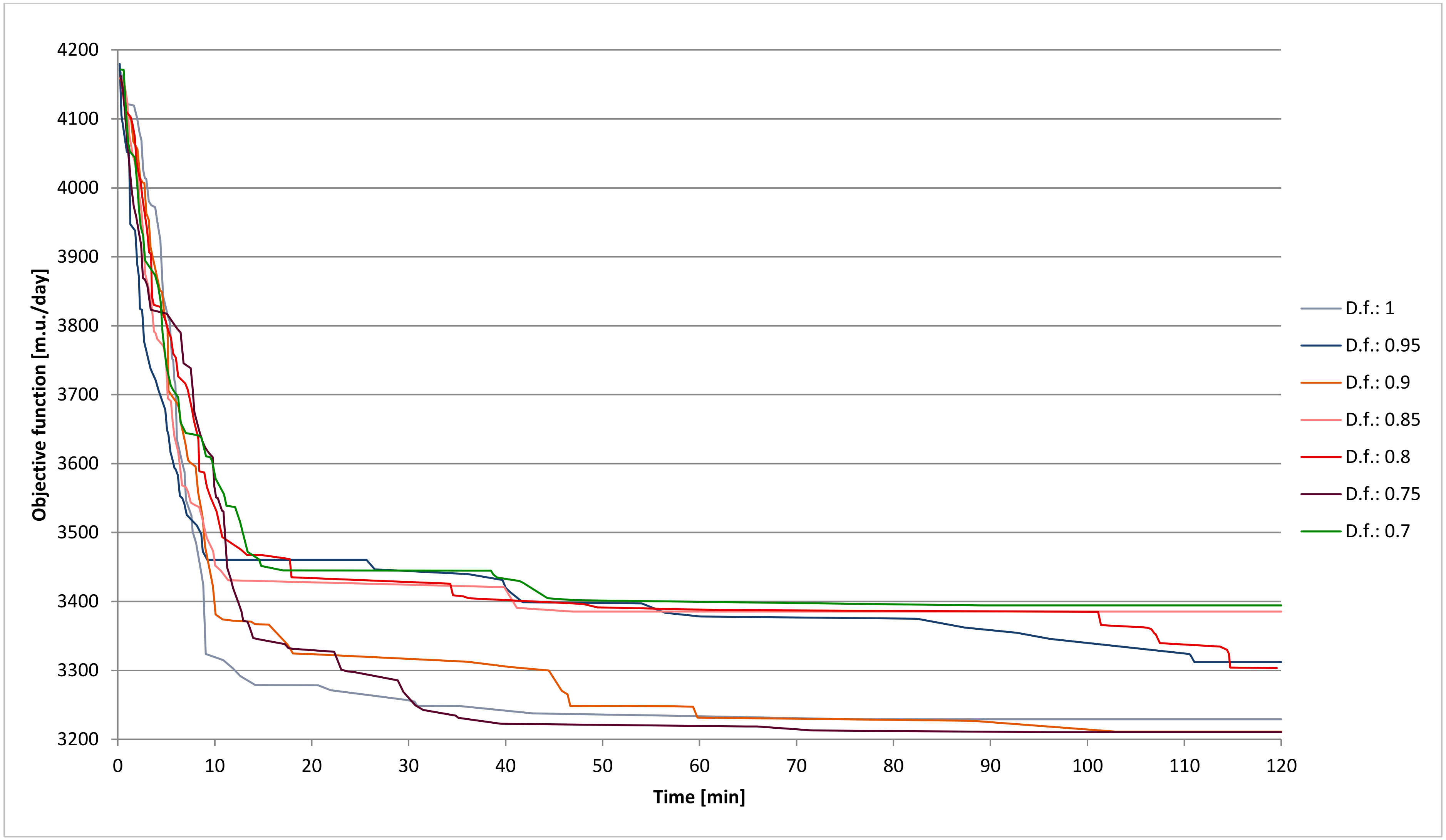


Figure 11.4: Iteration graph of the combinations of D.f. with Δs=100, ΔS=100

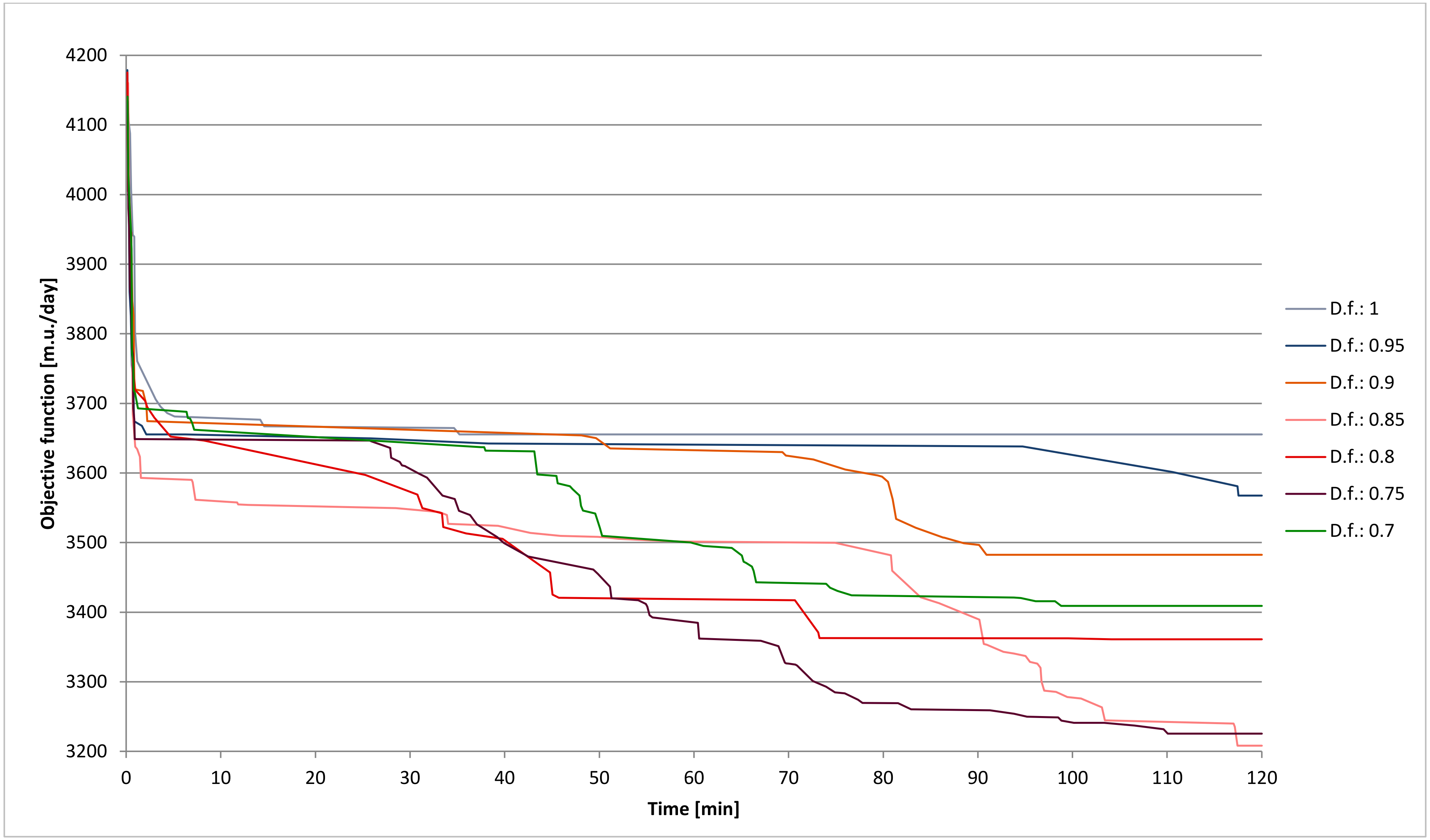


Figure 11.5: Iteration graph of the combinations of D.f. with Δs=500, ΔS=500

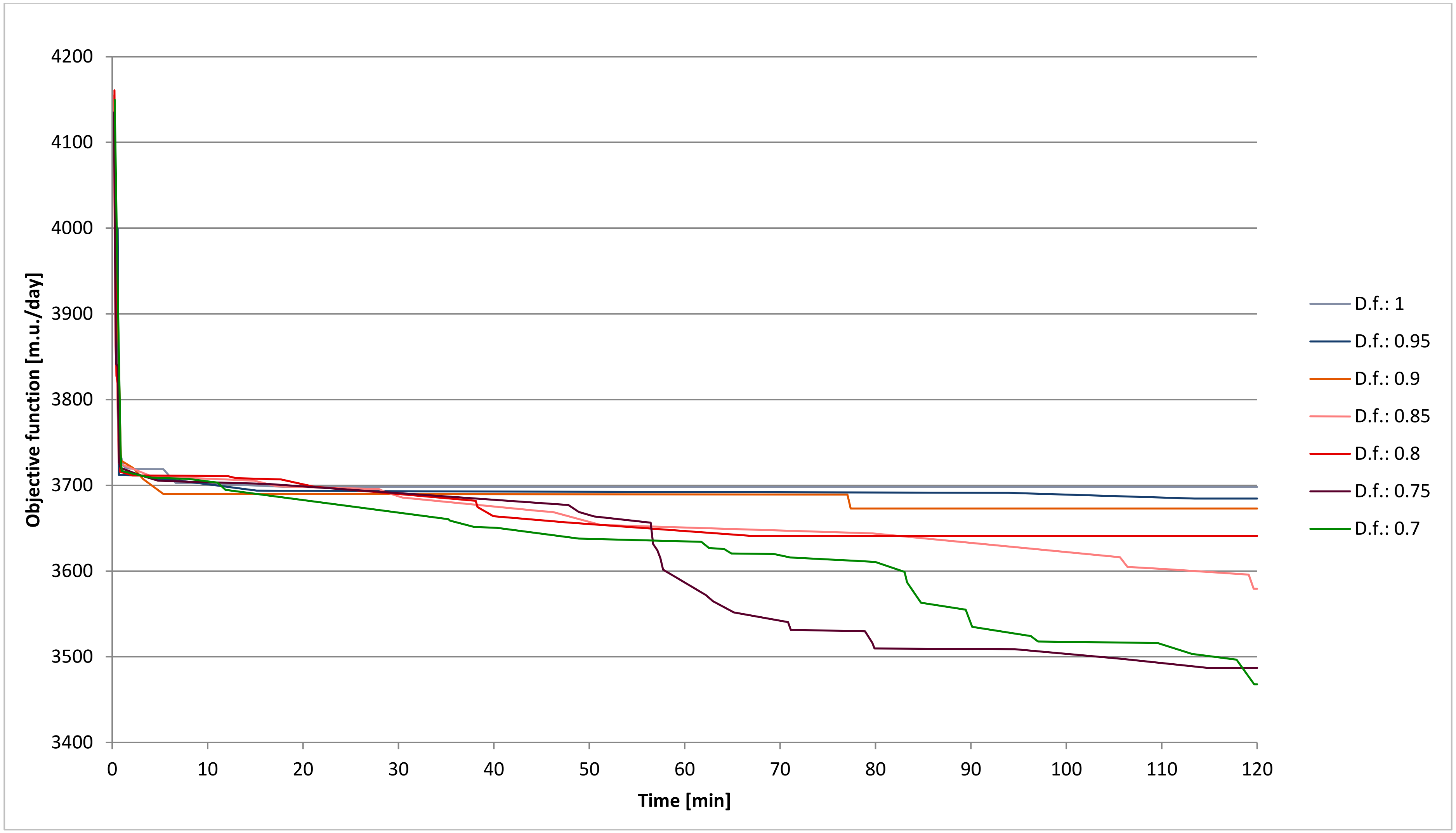


Figure 11.6: Iteration graph of the combinations of D.f. with Δs=1000, ΔS=1000

11.3 DTS, STS and DLS sensitivity analysis

11.3.1 (s,Q)

Dynamic TS	Static TS	DLS	Dynamic TS	Static TS	DLS
t*(s)	t*	t*	F*	F*	F*
2323.9	85.53	1829.3	2511.6	2506.93	2501.5
81.8	204.32	2859.5	2500.0	2508.01	2750.1
709.3	127	3018.3	2473.6	2508.18	2568.5
3531.3	537.7	435.0	2473.3	2490.89	2529.9
422.0	617	1791.8	2475.6	2485.87	2786.9
1905.6	210	1253.4	2445.7	2473.57	3158.6
1330.3	388	1357.7	2474.4	2503.52	2821.7
1253.7	467	313.8	2482.4	2503.77	2497.1
2361.1	171.48	0.0	2462.7	2462.71	3222.3
3531.3	341	1407.5	2473.3	2501.68	2492.3

11.3.2 (s,S)

Dynamic TS	Static TS	DLS	Dynamic TS	Static TS	DLS
t*(s)	t*	t*	F*	F*	F*
6936.0	2819.2	1198.3	3264.0	3212.2	3643.2
6074.0	2551.3	2094.5	3224.5	3224.8	6820.4
5698.2	4004.3	260.9	3339.8	3272.4	3536.2
7154.5	6772.0	3490.5	3302.9	3364.3	5487.1
6252.1	3021.4	5334.8	3230.3	3367.2	4908.5
6829.0	6775.2	1011.8	3327.6	3232.8	3539.7
6977.1	6986.8	366.7	3268.4	3278.5	3603.6
4285.2	5731.8	584.5	3199.0	3239.4	3696.9
6190.6	6342.3	1279.0	3299.2	3323.1	4496.5
7105.6	6785.8	2632.2	3319.6	3313.4	4165.3

11.4 Detailed cost of the results

11.4.1 Cost of the best solution (s,Q)

Solution of the (s,Q) case is found in the following table, the cost is the daily cost. The daily cost results from dividing the cost of all the days by the number of days.

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	20,27	66,23	50,50	26,97
Holding cost	[m.u.]	946,94	356,08	300,45	152,54
Launching cost	[m.u.]	177,35	132,95	115,85	77,35

11.4.2 Cost of the best solution (s,S)

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	28,22	184,66	52,09	92,01
Holding cost	[m.u.]	1014,26	407,63	575,42	376,11
Launching cost	[m.u.]	137,70	130,50	91,90	63,20

11.4.3 Cost for the several service levels (s,Q) case

SL=0.92			SL=0.97			SL=0.95		
Cost	SL	Daily cost	Cost	SL	Daily cost	Cost	SL	Daily cost
1161110	0.897	2322.22	1320640	0.976	2641.28	1233540	0.968	2467.08
1164340	0.906	2328.68	1293070	0.954	2586.14	1218310	0.951	2436.62
1180460	0.921	2360.92	1307650	0.959	2615.30	1189520	0.936	2379.04
1182770	0.883	2365.54	1345850	0.988	2691.70	1209520	0.951	2419.04
1186770	0.811	2373.54	1339050	0.981	2678.10	1209340	0.957	2418.68
1170200	0.913	2340.40	1317370	0.982	2634.74	1193330	0.942	2386.66
1163380	0.907	2326.76	1286130	0.929	2572.26	1220370	0.940	2440.74
1156710	0.907	2313.42	1300500	0.988	2601.00	1193690	0.955	2387.38
1175810	0.882	2351.62	1326770	0.976	2653.54	1206290	0.938	2412.58
1204550	0.901	2409.10	1312880	0.951	2625.76	1177530	0.945	2355.06
AVG	0.893	2349.220	AVG	0.968	2629.982	AVG	0.948	2410.288
Std Dev	0.030	27.570	Std Dev	0.018	36.273	Std Dev	0.010	31.793
IC	0.021	19.722	IC	0.013	25.948	IC	0.007	22.743
IC Min	0.871	2329.498	IC Min	0.955	2604.034	IC Min	0.942	2387.545
IC Max	0.914	2368.942	IC Max	0.982	2655.930	IC Max	0.955	2433.031

Distribution of the costs:

SL=0.92

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	26.71	75.36	70.76	53.87
Holding cost	[m.u.]	856.79	334.37	253.60	129.98
Launching cost	[m.u.]	181.50	136.05	129.05	80.85

SL=0.98

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	9.28	123.33	51.68	24.85
Holding cost	[m.u.]	1142.63	266.40	271.52	174.49
Launching cost	[m.u.]	206.70	138.35	121.50	81.75

11.4.1 Cost for the several service levels (s.S) case

SL=0.88			SL=0.97			SL=0.94		
Cost	SL	Daily cost	Cost	SL	Daily cost	Cost	SL	Daily cost
1567630	0.882	3135.26	1727060	0.981	3454.12	1620470	0.913	3240.94
1591030	0.888	3182.06	1702830	0.964	3405.66	1616120	0.960	3232.24
1569540	0.860	3139.08	1694430	0.960	3388.86	1640220	0.958	3280.44
1564230	0.892	3128.46	1719400	0.970	3438.80	1604380	0.952	3208.76
1552890	0.875	3105.78	1744500	0.957	3489.00	1621810	0.950	3243.62
1553010	0.890	3106.02	1719910	0.953	3439.82	1611990	0.935	3223.98
1558870	0.893	3117.74	1679340	0.970	3358.68	1573370	0.930	3146.74
1566300	0.908	3132.60	1725850	0.976	3451.70	1594970	0.928	3189.94
1594160	0.858	3188.32	1712360	0.962	3424.72	1608110	0.934	3216.22
1549150	0.854	3098.30	1708930	0.976	3417.86	1605010	0.934	3210.02
AVG	0.880	3133.36	AVG	0.967	3426.92	AVG	0.939	3219.29
Std Dev	0.017	29.04	Std Dev	0.009	34.87	Std Dev	0.014	33.65
IC	0.012	20.78	IC	0.006	24.94	IC	0.010	24.07
IC Min	0.868	3112.59	IC Min	0.961	3401.98	IC Min	0.929	3195.22
IC Max	0.892	3154.14	IC Max	0.973	3451.87	IC Max	0.949	3243.36

Distribution of the costs:

SL=0.88

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	77.59	264.96	80.13	112.16
Holding cost	[m.u.]	913.12	392.88	549.67	398.63
Launching cost	[m.u.]	135.10	125.80	87.40	59.25

SL=0.97

		Retailer	Wholesaler	Distributor	Factory
Backorder cost	[m.u.]	9.24	64.24	102.90	56.12
Holding cost	[m.u.]	1236.34	622.55	423.52	455.72
Launching cost	[m.u.]	137.90	131.80	92.45	63.40

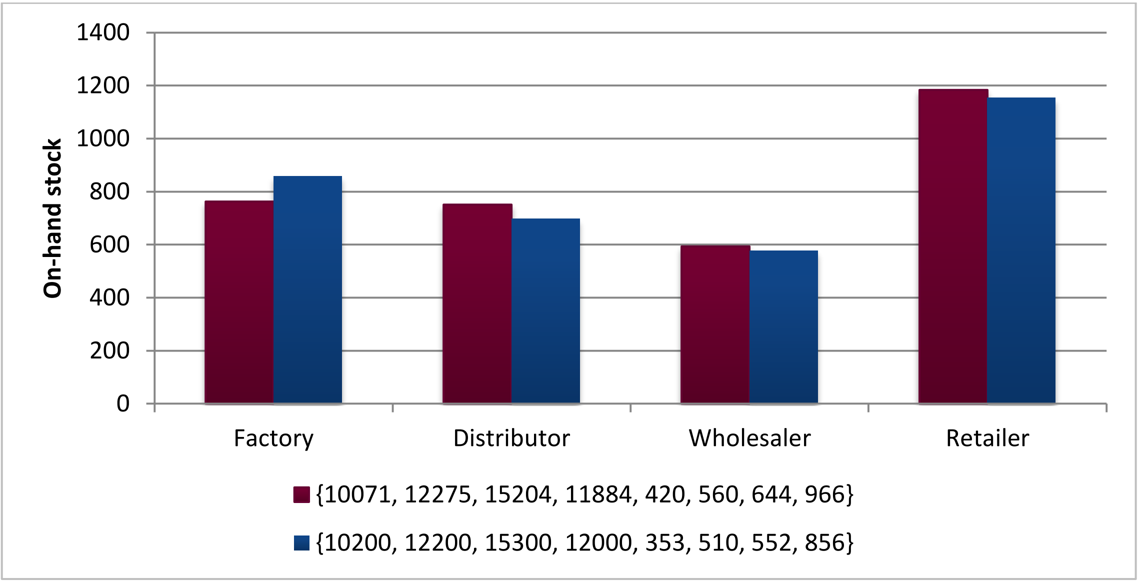
11.5 Other solutions found

A different solution was found using the tabu search. Other solutions different than the best are found, but the differences in the values of the solution are small so they are not considered. But a solution with a remarkable difference is found, that leads to a similar cost and on-hand inventory but has a different distribution of costs. This is expected as the solution has 8 variables, so a lot of combinations can be made and the optimal solution does not have to be unique. This solution found has these values (Table 11.1):

Table 11.1: Solution for the (s,Q) case II

		Factory	Distributor	Wholesaler	Retailer
s_i	[pieces]	12000	15300	12200	10200
Q_i	[pieces]	856	552	510	353

To compare this with the other solution a bar figure is made:



The two solutions, the one of the previous chapter and this new one have similar total cost. However, the length of the confidence interval is different, as the interval of the new solution includes the interval of the first one.

	First solution	New solution
Daily Total cost 95% C.I.	[2372.56 , 2402,41]	[2365.03 , 2428.01]
Service Level 95% C.I.	[0.937 , 0.957]	[0.926 , 0.952]