SUPPLY CHAIN MANAGEMENT IN ROLLING HORIZON FOR TAC/SCM

Luiz Aizemberg
luizaizemberg@gmail.com
Artur Alves Pessoa
artur@producao.uff.br
Eduardo Uchoa Barboza
uchoa@producao.uff.br
Tiago Neves
tneves@ic.uff.br
Guilherme Coutinho
gfcoutinho@gmail.com

Universidade Federal Fluminense, Departamento de Engenharia de Produção, Niterói, Brasil
Abril, 2009

RESUMO

No jogo Trading Agent Competition/Supply Chain Management (TAC/SCM), cada competidor é um agente de software autônomo controlando uma fábrica montadora de computadores pessoais. Dentre as atribuições de cada agente, encontram-se o escalonamento da produção e a negociação da compra de componentes e da venda de produtos. A cada rodada, o agente tem 15 segundos para terminar sua jogada. É declarado vencedor o agente que obtiver o maior lucro total. Neste artigo, definimos um problema de otimização a partir de uma estratégia de horizonte deslizante para o TAC/SCM. Para resolver o problema proposto, desenvolvemos um modelo de programação inteira mista (MIP) e uma heurística que utiliza a relaxação linear do MIP. Experimentos mostram que a heurística, quando integrada ao CPLEX, permite a geração de boas soluções viáveis em tempo hábil e que o agente proposto apresenta um desempenho aceitável.

PALAVRAS-CHAVE: Planejamento de Produção, Programação Linear Inteira, TAC/SCM

ABSTRACT

In the Trading Agent Competition/Supply Chain Management (TAC/SCM), each competitor is an autonomous software agent controlling a factory that assembles personal computers. Agent attributions include production scheduling, component acquisition and product selling. In each round, each agent must finish its play in up to 15 seconds. The agent having the largest total profit is declared the winner. In this paper, we define an optimization problem based on a rolling horizon strategy for the TAC/SCM. To solve the proposed problem, we develop a Mixed Integer Programming (MIP) model and a heuristic that uses the MIP’s linear relaxation. Our experiments show that the heuristic, when integrated to CPLEX, allows for obtaining good feasible solutions in time and that the performance of the proposed agent is acceptable.

KEYWORDS: Production Planning, Mixed-Integer Programming, TAC/SCM
1. Introduction

Supply chain management (SCM) is the management of a network of interconnected businesses involved in the ultimate provision of product or service packages required by end customers (Harland, 1996). It cares about all transportation and storage of raw materials, inventory capacities and costs and finished goods. The aim of SCM is the inventory reduction, but ensuring that no product will be missing when needed (Gomes et al., 2004). The development of techniques and tools to improve the management of supply chains contributes to improve their strategies and practices, leading to better decisions (Shim et al., 1999).

An interesting artifact in this scenario is the TAC/SCM game (see Sadeh et al, 2003 and Sardinha, 2005) that simulates a supply chain of computers and its components through a multi-agent platform. In this game, six agents compete for clients and for suppliers aiming at maximizing their own profit rate. On every simulated day, the agents have to deal with clients and suppliers, negotiating prices and delivery dates as well. At the end of the day, the agents must decide which clients to answer, what components to ask and from which suppliers. The agents are limited for their production capacity so that they can not produce computers unrestrictedly. In the end of the simulation, the agent that has the greatest value in its bank account is declared the winner.

1.1. Dealing with the problem

In this paper, we use a rolling horizon (Simpson, 1999) approach to devise an optimization problem that can be applied to the TAC/SCM game. Rolling horizon is a technique that solves part of the problem based in forecasts. Basically, this technique defines a sub horizon starting at the first period and ending T periods later, and calculates a solution for it. Under standard practice, the model is one period later reapplied covering periods t = 2, ..., T+1 with updated demand data over the rolled-forward T-period horizon, then over periods t = 3, ..., T+2, and so on, using fresh demand forecasts (Clark 2005). Once the optimization problem is well defined, we focus on finding good feasible solutions for it in up to 8 seconds, i.e. leaving 7 seconds for the remaining tasks required to complete each game round. Two optimization techniques are employed: a mixed-integer programming formulation and a heuristic based on its continuous relaxation. In both cases, the forecast data are regarded as deterministic. The results show that this kind of strategy applied quite well for the TAC/SCM game scenario.

1.2. Heuristic’s related work

The method used here is similar to the Relax-and-Fix proposed by Wolsey (1998). Relax-and-Fix consists of \( n \) steps. In the first step, it solves a subproblem where the integralities of some integer variables are dropped. Then, in the \( i \)th step, for \( i = 2, \ldots, n \), it fixes the variables that were considered as integer in the previous step to their current values and selects a new subset of non-fixed integer variables to be regarded as integers. After the last step, if no intermediate subproblem was infeasible, a feasible solution is obtained.

Several authors have proposed variations of the Relax-and-Fix heuristic in the literature. Here, we mention only some of them, which are applied to production problems over a time horizon. In Pessoa (2006), a decomposition heuristic was proposed for the problem of producing iron ore pellets in multiple plants. The heuristic consists of fixing continuous variables based on the continuous relaxations of a global MIP so that the resulting problem becomes independent for each plant. Then, each subproblem is solved through another (local) MIP, generating a feasible solution to the global problem. Such solution is further improved using the global MIP formulation. Araujo et al (2007) combined the Relax-and-Fix heuristic with several other heuristics to solve the resulting subproblems, such as local search descent and simulated annealing. For the problem that we address in this paper, we observe that feeding the standard CPLEX branch-and-cut with a feasible solution calculated in a suitable way is enough to obtain a high-quality solution at the end of the available running time.

1.3. Paper organization
This paper is organized as follows: in Section 2 TAC/SCM game is explained briefly, in section 3 the equations of the mathematical formulation are outlined and explained. Section 4 exposes the proposed heuristic. The fifth section shows the results obtained. Section 6 concludes this work.

2. TAC/SCM

The game consists of 6 competitors that need to maximize their final bank accounts. In each round, they compete with the opponents for customer orders through closed envelop auctions. Customers start sending requests for quote (RFQs) to the agents and wait for their offers. An RFQ completely determine the configuration of the computer to be delivered and also gives the quantity and the maximum price accepted by the customer. The lowest price offer wins the customer order. Once the order is made, the player that receives it must deliver the specified computers in the specified quantity, and meeting the specified due date. If such due date is not satisfied, the player pays a high penalty per day until the 5th late day, when the order is canceled. To assemble the different types of computers ordered from customers, players need to buy components from suppliers through similar negotiation. For each computer configuration, the required bill of materials is given, where some components are required by more than one computer type and/or provided by more than one supplier. The players start sending RFQs to suppliers with the highest price they want to pay, due date and quantity. Before sending offers, suppliers can adjust quantity and due date of received requests from agents. If the player’s highest price is not attractive, it will not receive any offer. With the components in stock, the player sends requests for its factory to assemble the computers, regarding the limited capacity of the assembly line. This routine is repeated upon every game round (referred to as a game day), where a number of factory cycles is available to assemble computers. For each computer type, the number of cycles required to assemble each unit is also given. The quantity of RFQs from customers and the suppliers’ production capacities are generated from known probabilistic distributions. Bank loans are unlimited. If the bank account balance is negative, the competitor pays interest. If positive, it receives.

3. Agent Model

3.1. Problem Description

The problem defined below aims to optimize the behavior of an agent for TAC/SCM game for one day, but considering a longer time horizon. As demands, we consider only the customer RFQs received in the current day (no forecasts for future RFQs are used) and the customer orders made until that day that are still not delivered or canceled. Since each RFQ due date is at most 12 days later than the day it was received, there is no need to extend the horizon for more than 17 days (12 days + 5 days until it is canceled).

In addition, the problem uses the following assumptions:

1. We are given the current state of the player in the given day including the bank account balance, the stock levels for all computer types and components, the incoming components (ordered to suppliers but still not received), the pending orders, their prices, quantities and due dates.
2. Any component is available to the agent after a minimum number days past from the current one (currently, we use 4 days: 2 for trading, 1 for production by the supplier and 1 for delivery) for a given price (currently, we use a rough estimation of that price). However, the quantity of components of each type to be received in each day is limited.
3. Any offer to the customer is assumed to be responded with an order if a given price is met (currently, we use the maximum price that the customer accepts to pay). As a result, orders
and RFQs are not distinguished from each other, except for the penalty. The penalty for an RFQ can be avoided if the player decides not to generate the corresponding offer.

4. The objective of the model is to maximize the bank account in the last day of the considered horizon.

Let us refer to the optimization problem describe above as the *Rolling Horizon SCM Problem* (RHSCMP). Next, we give a MIP formulation for the RHSCMP.

3.2. *Constants*
3.3. Variables

$$CI_j$$ - Amount of time (or cycles) required to assemble a PC of type $$j$$.

$$Q_N$$ - Quantity of PC's of the order or RFQ $$N$$.

$$Type_N$$ - Type of PC's of the order or RFQ $$N$$.

$$Delivery_N$$ - Delivery day of the PC's of the order or RFQ $$N$$.

$$IPP$$ - First PC index.

$$IUP$$ - Last PC index.

$$IUC$$ - Last component Index. $$IPP = IUC + 1$$.

$$UD$$ - Last day of the time horizon.

$$P_j$$ - Purchase price of the component $$j$$ (for $$j = 0, ... , IUC$$) or sale price of the PC $$j$$ (for $$j = IPP, ... , IUP$$).

$$\alpha$$ - Annual bank interest rate received when the bank account is positive.

$$\alpha^-'$$ - Annual bank interest rate paid when the bank account is negative.

$$E$$ - Number of days in the game.

$$S$$ - Annual storage cost rate.

$$A$$ - Down payment due on placement of supplier order.

$$\Psi$$ - Daily penalty paid for delay in orders delivery. It is paid from the first to the last day allowed after the deadline.

$$U_{j,k}$$ - Amount of component $$j$$ required to produce a PC of type $$k$$.

$$n$$ - Total number of orders and RFQs.

$$lastRFQ$$ - Total number of RFQs. RFQs are indexed from 1 to $$lastRFQ$$ and orders from $$lastRFQ + 1$$ to $$n$$.

$$NMD$$ - Minimum number of days required between the completion of the player's RFQ to the supplier and the delivery of the corresponding components.

$$AM$$ - Maximum delay allowed (days).

$$QCD$$ - Number of daily production cycles.

$$EI_j$$ - Initial stock of component or PC of type $$j$$.

$$SBI$$ - Initial bank account.

$$QRF_{d,j}$$ - Quantity of components of type $$j$$ to be received by the player in the day $$d$$.

$$MRF$$ - Maximum quantity of components of each type that can be received in each day.

$$PENALTY_N = \Psi \times Q_N \times P_{Type_N}$$.
$QEB_{d,N}$ - Binary variable equal to 1 when the player will deliver the order or RFQ $N$ in the day $d$.

$QR_{d,j}$ - Integer variable representing the amount of components of type $j$ that the players will receive in the day $d$. It means that the player must do the corresponding RFQ to the supplier at least $NMD$ days before $d$.

$X_{d,j}$ - Integer variable representing the quantity of PC's of type $j$ to be produced in the day $d$.

$W_{d,j}$ - Integer variable representing the inventory of component or PC of type $j$ in the day $d$.

$b_d$ - Continuous variable representing the bank account in the day $d$.

$BIN_N$ - Binary variable equal to 1 when the player will not generate any offer corresponding to the RFQ $N$. There is no $BIN$ variable associated to an order.

### 3.4. Formulation

**Maximize** $b_{UD}$

**Subject to**: 

Constraints that control the inventory balance along the time horizon:

$$W_{(d+1),j} = W_{d,j} + QR_{(d-1),j} - U_{j,k} \times X_{d,k} \quad d = 0 \ldots UD - 1, j = 1, \ldots, IUC, k = IPP, \ldots, IUP \quad (1)$$

$$W_{(d+1),j} = W_{d,j} - \sum_{N=1}^{n} Q_N \times QEB_{d,N} + X_{(d-1),j} \quad d = 0, \ldots, UD - 1, j = IPP, \ldots, IUP \quad (2)$$

$$W_{0,j} = EI_j \quad j = 1, \ldots, IUP \quad (3)$$

Constraints ensuring that each delivery occurs at most once:

$$\sum_{d=0}^{UD} QEB_{d,N} + BIN_N = 1 \quad N = 1, \ldots, lastRFQ \quad (4)$$

$$\sum_{d=0}^{UD} QEB_{d,N} \leq 1 \quad N = lastRFQ + 1, \ldots, n \quad (5)$$

Constraints that limit the daily production:

$$\sum_{j=IPP}^{IUP} CI_j \times X_{d,j} \leq QCD \quad d = 0, \ldots, UD \quad (6)$$

Constraints imposing that $b_d^+ = 0$ and $b_d^- = b_d$ when $b_d < 0$ and that $b_d^+ = b_d$ and $b_d^- = 0$ when $b_d > 0$:

$$b_d = b_d^+ - b_d^- \quad b_d^+, b_d^- \geq 0 \quad d = 0, \ldots, UD \quad (7)$$
Constraints that control the bank account balance along the time horizon:

\[ b_0 = SBI \]

\[ b_{(d+1)} = (1 + \frac{a}{E}) \times b_d - (1 + \frac{a^*}{E}) \times b_d + \sum_{N=1}^{n} QEB_{d,N} \times Q_N \times P_j - \sum_{j=1}^{IP} S \times P_j \times W_{d,j} \]

\[ - \sum_{j=1}^{IUC} A \times P_j \times QR_{(d,NMD),j} - \sum_{j=1}^{IUC} (1 - A) \times P_j \times QR_{d,j} \]

\[ - \sum_{N=1}^{lastRFQ} \text{Penalty}_N \times (1 - \sum_{x=0}^{d} QEB_{r,N} - BIN_N) \]

\[ - \sum_{N=lastRFQ+1}^{NMD} \text{Penalty}_N \times (1 - \sum_{x=0}^{d} QEB_{r,N}) \]

\[ d = 0, \ldots , UD - 1 \]  

Constraints on the supplier capacity:

\[ QR_{d,j} = QRF_{d,j} \quad d = 0, \ldots , NMD \quad j = 1, \ldots , IUC \]  

\[ QR_{d,j} \geq QRF_{d,j} \quad d = NMD + 1, \ldots , UD \quad j = 1, \ldots , IUC \]  

\[ QR_{d,j} \leq MRF \quad d = NMD + 1, \ldots , UD \quad j = 1, \ldots , IUC \]  

Non-negativity constraints:

\[ W_{d,j} \times X_{d,j} \geq 0 \quad d = 0, \ldots , UD \quad j = 1, \ldots , IUP \]  

\[ QR_{d,j} \geq 0 \quad d = 0, \ldots , UD \quad j = 1, \ldots , IUC \]  

By our experiments, the previous formulation gives strong lower bounds for the RHSCMP. However, the standard CPLEX branch-and-cut algorithm is unable to find feasible solutions for many instances generated during the game in up to 8 seconds. This observation motivates the development of a specific Relax-and-Fix like heuristic for this problem that we describe next.

### 4. Heuristic

Our heuristic can be described by the following steps:

1. Calculate the linear relaxation of the model.
2. Select in the relaxed solution for the \( QEB \) variables equal to 1.
3. Solve the same MIP formulation with all values of \( QEB \) variables fixed. The \( QEB \) variables selected in the previous step are set to 1 and the remaining ones to zero.

Note that the criterion used in the Step 2 to select \( QEB \) variables is very restrictive but it has the advantage of ensuring that the resulting MIP solve in the Step 3 is always feasible. Moreover, our experiments showed that some orders and RFQs have associated \( QEB \) variables selected in the Step 2. Then, the obtained solution is used to feed the CPLEX branch-and-cut algorithm through the heuristic callback, where it is improved by the Relaxation Induced Neighborhood Search (RINS) implemented by the software. Although the solution obtained by our heuristic is typically poor, we observe that it is an excellent starting point for the CPLEX RINS.

### 5. Experimental Results

In this section, we compare the results of solving the MIP model described in the Section 3, using CPLEX with and without our heuristic. We observe different results when the CPLEX solver
uses the feasible solution given by the heuristic at the beginning and when it needs to find a feasible solution. The RHSCMP instances were collected during a TAC/SCM game with our player against 5 example (dummy) agents available in the TAC/SCM platform. The instances have about 100 clients RFQs, 35 customer orders and 50 supplier offers per day. Clients can order about 16 different types of PCs, and these PCs can be assembled with 10 different components, 4 components per PC. The resulting MIP has about 1240 restrictions and 4220 variables.

The results are analyzed based on the relative gap between the best feasible solution ($LB$) and the best upper bound ($UB$) obtained by CPLEX. However, we observe that this relative gap may be meaningless when it directly uses the absolute value of the objective function. This is true because the initial bank account balance is different of each instance. Moreover, some instances correspond to days where the player is already compromised with the delivery of many orders while other instances may contain no initially pending orders. In order to obtain an objective value that better measure the success of the solution on increasing the bank account balance (or not letting it decrease too much), we define a reference value for the objective function of each instance. This reference value (denoted by $REF$) is the final bank account balance that would be obtained if the players did not request any production and did not deliver any order or RFQ. Of course, in the case of pending orders, it would pay the corresponding penalties for canceling them. As a result, the gap is calculated as

$$\text{gap} = \frac{UB - LB}{LB - REF} \times 100\%.$$

5.1. Gaps without Heuristic

In the graph of Figure 1, each dot represents the gap obtained for the instance generated in a game day. The values equal to 100% means that the CPLEX solver has found no feasible solution in 8 seconds. These results motivated the development of the heuristic.

Analyzing the graph, we see many gaps between 0% and 0.1%, a small amount between 0.1% and 10%, and many instances where solver could not find a feasible solution in time. More precisely, 79 days.

Another way to see the results is given by Table 1. In this table, the 79 days in which no feasible solution was found were discarded.
By Table 1, we note that the best feasible solution found by the CPLEX solver is usually very good on the instances where such solution can be found.

5.2. Gaps with Heuristic

The graph of Figure 2 is similar to that of Figure 1 but considering the feasible solutions found with the heuristic embedded in the CPLEX branch-and-cut algorithm. In this graph, we observe many gaps between 0% and 0.1%, a considerable amount between 0.1% and 1%, some gaps above 1%, and two above 10%. However, feasible solutions have been found for all instances.

In the same fashion, Table 2 is similar to Table 1 but considering the feasible solutions found with the heuristic embedded in the CPLEX branch-and-cut algorithm. Although the maximum gap is high, we observe that the average gap is very low, and only slightly higher than that without the heuristic. The low standard deviation shows that the heuristic is very robust.

<table>
<thead>
<tr>
<th>Gaps without Heuristic</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.003%</td>
<td>10.701%</td>
<td>0.322%</td>
<td>0.013527607</td>
</tr>
</tbody>
</table>

Table 1 - Gaps without the heuristic

<table>
<thead>
<tr>
<th>Gaps with Heuristic</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.001%</td>
<td>39.204%</td>
<td>0.414%</td>
<td>0.029043</td>
</tr>
</tbody>
</table>

Table 2 - Gaps with the heuristic
5.3. With Heuristic x Without Heuristic

The graphs and tables of Subsection 5.1 and 5.2 may suggest that the method without the solution leads to a better solution quality in the cases where such solution can be found. However, this is not true. The lower average gap is explained by the fact that no feasible solution is found for the hard instances. So, this average considers only the easy instances. To get a better comparison between the two methods, we show the graph of Figure 3 where the gaps obtained with and without the heuristic are represented side by side for the instances generated in game days that are multiple of 10. The values of 100% mean that no feasible solution was achieved in 8 available seconds.

![Graph showing comparison between With Heuristic and Without Heuristic](image)

**Figure 3 - With Heuristic x Without Heuristic**

In this figure, we see that the CPLEX solver without the heuristic failed on 5 occasions. In other days, the solver with the heuristic was better on 15 instances, and worse on 6. A tie occurred on 1 instance.

<table>
<thead>
<tr>
<th>With Heuristic x Without Heuristic</th>
<th>Greater difference in favor of heuristic</th>
<th>Greater difference against heuristic</th>
<th>Average*</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.570%</td>
<td>2.811%</td>
<td>0.241%</td>
<td></td>
</tr>
</tbody>
</table>

*values in favor are positive and against are negative.

**Table 3 - With Heuristic x Without Heuristic**

Table 3 contains information about the difference between the gaps with and without the heuristic, for all instances but the 79 ones in which no feasible solution was found without the heuristic. The table shows that the mean difference is negative, that is, the sum of the differences in favor of the heuristic is greater than the sum that of the differences against the heuristic.

5.4. Result of a TAC-SCM test game against 5 dummy agents
Table 4 shows that the player that used our optimization method has finished a test game with 40 million of bank account balance, and the second best player obtained only 11 million. The result was obtained using the heuristic.

<table>
<thead>
<tr>
<th>Player</th>
<th>Revenue</th>
<th>Interest</th>
<th>Costs</th>
<th>Margin 1</th>
<th>Margin 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Material</td>
<td>Storage</td>
<td>Penalty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logis</td>
<td>100 513</td>
<td>603 625</td>
<td></td>
<td>55 463</td>
<td>354 362</td>
<td>5 225 585</td>
</tr>
<tr>
<td>Dummy-5</td>
<td>86 804</td>
<td>59 039</td>
<td></td>
<td>74 634</td>
<td>730 830</td>
<td>153 625</td>
</tr>
<tr>
<td>Dummy</td>
<td>88 926</td>
<td>40 088</td>
<td></td>
<td>77 412</td>
<td>689 538</td>
<td>331 515</td>
</tr>
<tr>
<td>Dummy-3</td>
<td>89 581</td>
<td>37 574</td>
<td></td>
<td>78 259</td>
<td>718 387</td>
<td>300 865</td>
</tr>
<tr>
<td>Dummy-2</td>
<td>86 535</td>
<td>6 223</td>
<td></td>
<td>76 604</td>
<td>662 450</td>
<td>741 888</td>
</tr>
<tr>
<td>Dummy-4</td>
<td>85 755</td>
<td>3 617</td>
<td></td>
<td>75 977</td>
<td>702 894</td>
<td>625 420</td>
</tr>
</tbody>
</table>

Margin 1 is the margin excluding bank interest and penalties while Margin 2 includes bank interest and penalties.

Table 1 – Game Log

In this table, the Player Column shows the name of the players, the Revenue Column shows the money earned from sale, the Interest Column shows the gain/loss on bank interest, the Material Column shows the total cost of the acquired components, the Storage Column shows the total cost of PCs and components, the Penalty Columns show the total cost of late delivery penalties and its percentage of the total cost, the Margin 1 Column shows the profit margin excluding bank interest and penalties, the Margin 2 Column shows the profit margin including theses items, and the Result Column shows the final bank account.

The graph of Figure 4 shows the evolution of the bank account of our player at the same game.

Figure 1 – Bank account with heuristic

6. Conclusion

In this paper, we studied an optimization problem applied to the TAC-SCM game that follows a rolling horizon approach. Our focus was designing a MIP based method to find good feasible solutions to this problem in a limited amount of time. This feature is a requirement for the TAC-SCM game since each player must complete a game round in up to 15 seconds (and we must leave at least 7 CPU seconds for other processes). The most important result of this paper is the...
observation that fixing the values of properly selected binary variables in the MIP model leads to feasible solutions that, although poor in terms of objective function, are excellent starting points for the CPLEX branch-and-cut algorithm. Our experiment showed a clear difference between the performances of CPLEX with and without the heuristic. It is worth mentioning that this difference also translates into a different performance of the player in the game. Without the heuristic, the bank account balance is typically 25% smaller than with the heuristic, when playing against dummy agents.

Although the current player has a good performance when playing against dummy agents, it is still not competitive when playing against more aggressive players. This occurs because our optimization model is a short term model. No long term strategy is implemented. As future work, we intend to include algorithms that implement such strategies and feed our model with guidelines derived from them.

Acknowledgements

This work is part of the research project "MAS-SCM: Protótipo de Ambiente Integrado para Soluções Logísticas com Uso de Multi-Agentes" financed by Petrobras.

AAP was partially financed by CNPq grants 301175/2006-3.

7. References