Dissecting and Understanding Supply Chains through Simulation: an Agent-based Approach

Gian Paolo Jesi[1] and Guido Fioretti[1]

University of Bologna
{jesi, fioretti}@cs.unibo.it

Abstract. The increasing interest about sustainability in supply chains highlights the influence of economic, social and environmental factors. The interaction of these factors generates an emerging complexity that is far beyond the 'sum of the parts'. This is why new tools are required to deeper understand and manage this complexity.
In this ongoing work, we focus on supply chain; in particular, as our ultimate goal, we focus on food mass purchasing consortium (MPC) market because of the availability of literature and data on this specific topic.
The main need of chain actors is to maintain competitive advantage. In this scenario, the typical questions arising in this environment are the following. What are the market opportunities for each actor position? Could we improve the chain performance by a more clever exploitation of the available information? How much information should be shared among actors?
We propose an agent based modeling (ABM) approach to study and identify the supply chain weaknesses by modeling its structural evolution over time. Our approach can provide answers to the previous questions. We believe this approach is promising since it could represent a valuable tool in (i) understanding supply chain processes and (ii) in testing social, economical and political strategies in order to achieve better sustainability models.

1 Introduction

Today's modern business is becoming more and more interconnected. Supply chains reflect this trend since they involve every aspect of the production and provision of goods and services.
They link all the actors involved in the process: starting from the producers, down to the distributors, to wholesalers, to retailers and finally to end consumers. A simple example of supply chain is summarized by the schema in Figure 1.
In general, we can think of a supply chain as a network of interconnected set of different members and relationships that are in a constant state of flux. On one hand, goods flow down the network from producers to consumers in response of the information perceived about their demand, on the other hand the information (orders) flows up to the network structure from end consumers to producers.

Supply chains can be considered as Complex Adaptive Systems (CAS), since their macro-level behavior can be mapped to the individual decision rules of each actor in the chain. Even a simple supply chain setup is very sensitive to perturbations that can lead to complex behaviors that are difficult to understand, manage and control. The tendency of a supply chain to exhibit unstable behavior is well known [1]. This behavior is characterized by (a) *oscillations* of orders and inventory size, (b) *amplifications* of the demand coming from the lower levels and (c) *phase lag*: the order rate tends to peak as orders flow up in the chain.

These issues can arise even in the smallest setup, therefore chaotic and unstable behavior is not a consequence of the size of the network but it is generated by the complex interaction among the participants and by the limit of the local decision rules that are just based on locally available information [2, 3].

These issues are the norm in real world supply chains and their inner causes are non-linear and are very hard to detect even for experienced managers. We believe new tools are required to tackle the problem of understanding and to better manage supply chains. We propose a dynamic supply chain model based on our simulation platform: *Agents and Complexity in Python* (ACP).

Supply chain participants can be considered as *social agents*, since they need to interact together in order to sort out many concurrent activities such as: seeking sources for their supply, finding outlets for their goods, negotiating transactions and share information. The impact value of information sharing in supply chain is tremendous. Modern IT infrastructures make sharing information a reality, but it is important to understand *what and how to share* in order to get a tangible benefit. The risk here is just to increase costs (e.g., integration, maintenance, ...).

However, a well designed information sharing can make the agents of the chain less and less isolated and more part of connected ecosystem (e.g., holistic view [4]). For example, sharing data allows to rely less on forecasting to make decisions and makes the chain as a whole less sensitive to fluctuations and unmanageable behaviors.

We have also to consider that business relations among participants does not last forever, since every party is constantly seeking for the best partners to play with. This makes the chain network dynamic also at the structure level. Understanding how the structure evolves and why, lead to the possibility to take action and possibly prevent that many participants will quit the business especially in a crisis period or when too aggressive market dynamics take the lead.

In this paper, we present a novel supply chain simulation model based on our agent-based simulation platform [5]. It is still an ongoing study that is leading to the crystallization of tools to deeply understand the dynamics of the supply chain. The model tackles the problem of understanding the value and the impact of the information shared and the evolution of the network structure.

This paper is organized as follows. We first describe our methodology and we introduce ACP, then we introduce our specific model aimed to supply chains. The next section shows results regarding our test scenario.

Finally, the last section draws some conclusions and present the future steps of our ongoing work.

2 Methodology

We believe simulation is a valuable approach when dealing with complex scenarios. By modeling a virtual or synthetic representation of the subject it is possible to track and understand deeper the dynamics behind it. In addition, it is possible to recreate specific conditions that are supposed to have generated issues and verify them. Simulation becomes a sort of laboratory where different *what if* scenarios can be explored and compared; at the same time business strategies can be adapted to deal better with the new conditions. Many options can be explored simultaneously, making simulation an desirable tool for the management.
In order to exploit the best benefit from a enterprise virtual model (e.g., production process, business plan, supply chain, ...), the main requirement is an active collaboration between the enterprise and the simulation experts. Only a synergic effort will lead to a realistic model and this represents the *most time consuming activity* required in order to apply the model with profit.
Our simulation platform ACP [5, 6], is aimed to address complex, asynchronous events that represent the interaction among the participants in the virtual model. Its *agent based* nature makes it particularly suited for bottom-up design.
In ABM, the actors or agents of the model are software objects that represent real world entities, such as: poeple (e.g., managers or workers), offices, orders, machines, sub-companies, factories, customers or any other entity we can think of in the domain we are addressing. One of the advantages of this approach is that it is usually very intuitive to come up with a first draft of the model (i.e., at least form the structural point of view).
Agent behaviors express the *actual actions each entity does* and not what it is supposed to do! It seems a subtle difference, but it is extremely important to understand the ABM spirit. In order to correctly express the entity behavior and implementing a non-biased bottom-up approach, it is crucial to get in touch with the company people and their processes. It is not unusual to find that the management is unaware of common practices and procedures that are normally carried out. Ignoring these, would lead to a faulty virtual model.
The main properties of an agent are in general the following: (a) has a limited view of the environment, (b) can communicate with other agents and exchange information (i.e., through event or messages) , (c) has its own behavior that can change according to its will or to external factor, (d) it is capable to perform one or more tasks and each task takes time (> 0). Many definitions of ABS can be found in literature, but all are essentially equivalent to the one provided.
ACP mixes features form the discrete event simulation area in order to efficiently manage the crowd of events that the agent interactions generate [6].

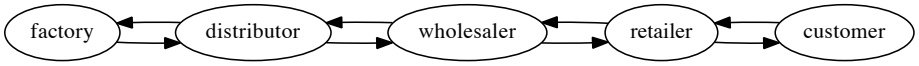


Fig. 1. A simple in-line supply chain. Five distinct stages are shown. The information can flow in both directions. Orders flow UP_{Stream}: from customer to factories, while supplies (or goods) flow $DOWN_{Stream}$: from factories to customers.

3 System model

On top our simulation engine (ACP), we developed a specific model to track the dynamics of supply chains. Essentially, ACP manages the agents that populate the simulation model by generating their instances, wiring them in a network structure and scheduling their behavior.

The link relations between agents are defined in a graph structure - AgentGraph - which defines the network topology and - in other words - defines a 'who knows whom' relation in the network. The simulator reads a graph description from a file where each node is node corresponds to a distinct agent instance.

Each agent is represented by a unique name or ID in the system. In addition, the graph description is enriched by a set of attributes defining the agent characteristics, such as its type, which corresponds to a specific class implementing the desired features of the real-world corresponding item. The actual file format may vary[1].

Each agent can communicate with its direct neighbors. For the supply chain particular network, we consider the graph undirected.

ACP is a message (event) driven simulator, therefore the behavior of an agent is run when a corresponding message is scheduled at a specific time. At the beginning of the simulation, ACP generates a message to trigger each agent behavior. Then each agent is free to schedule its next run according to its will. Usually, the next schedule is transparently managed by the simulator facilities, apart from very rare and specific cases.

As dictated by ABM agents are also reactive and can also react to specific messages sent by other agents.

Our model is an extension of the well known Sterman's *Beer Game* (BG) [7]. This choice is motivated by the fact that the BG is very well studied and simple. However, its simplicity comes not at the expense of expressiveness. In fact, the BG is the simplest model to capture the complex, chaotic nature of the interaction among supply chain actors. A ramp-up in customer demand causes hysteric fluctuations of over-ordering and under-ordering that can hardly be stabilized by each individual agent.

In its simplest form a supply chain is a sequence of steps (e.g., from manufacturing to distribution, wholesaling and retailing), as shown in Figure

[1] Our simulations adopted the open-source GraphViz format because of the simplicity with which it can be extended.

1. The steps exchange information such as goods and orders. In particular, goods flow $DOWN_{Stream}$ (e.g., from production to customers) and orders flow UP_{Stream} (e.g., from customers to production).
The basic idea of the interactions in the BG is the following:

1. at the lowest level, customers place orders UP_{Stream}
2. retailers fill the order request immediately if they have enough stock; otherwise, the customer request is put in *backorder* and will be filled when stock is available
3. retailers receive shipments from UP_{Stream} (e.g., wholesalers) according to the orders they placed previously. They can decide how much to order from the UP_{Stream} according to their behavior (i.e., *ordering rule*). The decision is the result of a forecasting function based on locally available data about demand.
4. steps 2 and 3 hold for every other agent type in the chain. The only exception is represented by the factory agent, which decides how much to produce instead of placing orders.

In the real world, it is obvious that there are multiple entities at the same stage of the chain: a retailer will have multiple sources from which it will replenish its stocks, and this is true at any stage of the chain. This is why we consider Supply Chain Networks (SCN), where each agent is connected to one or more agents from the upper stage.
Finally, we consider that also the interrelationships (i.e., the graph links) between agents are dynamic and are constantly in flux. Conditions evolve and change over time closing old, no-longer profitable links and allowing the creation of new ones. We added this feature to the static network chain model, allowing to monitor the structural evolution of the chain. This is our main contribution and extension to the classic BG.
The notion that makes a link valuable is the length of the backorder queue. Having a neighbor with a short queue means a responsive and reactive neighbor, while a long queue means long time to wait and difficulties to interact with. Each agent tends to close the link to upstream neighbors showing a long queue after a threshold δ.
In addition, each agent regularly search for a new UP_{Stream} companion. The upper limit to the number of neighbors is given by the number of agents belonging to the set of the next stage. For example, a customer agent can have all the retailer agents as UP_{Stream} neighbors. A stricter limit can be expressed in the simulation model, but its applicability can be considered for large networks.
It is not allowed for an agent belonging to a type (e.g., retailer) to skip a step of the hierarchy and to wire a relation with - for example - a distributor agent. This would be a desirable feature when analyzing the evolution of firms, but it is not adopted and discussed in this work.

4 Algorithm

A supply chain agent keeps track of the following main parameters:

- $Inventory_t$: current inventory level at time t
- $Inventory_{desired}$: desired inventory level

- $Pipeline_t$: current number of items in the pipeline (i.e., shipments in transit from UP_{Stream} agents and waiting orders placed by $DOWN_{Stream}$ agents) at time t
- $Pipeline_{desired}$: desired level of items in the pipeline
- $Demand_t$: current demand at time t
- $expDemand_t$: expected demand at time t

$$
\begin{aligned}
Demand &= OrderReceived \\
Stock_t &= Stock_{t-1} + SupplyReceived_t - SupplySent_t \\
Backorder_t &= Backorder_{t-1} + OrderReceived_t - SupplySent_t \\
Inventory_t &= Stock_t - Backorder_t \\
Pipeline_t &= Pipeline_{t-1} + OrderSent_t - SupplyReceived_t
\end{aligned}
$$

In general, we consider these parameters as *public*, meaning that each direct neighbor has the access to this information.
The agent **goal** is twofold: (i) *achieve the desired inventory level* and (ii) the *desired pipeline level*. The former ensures to be able to keep up with demand and the latter minimizes the delay in placing the order and receiving the corresponding supply.
The actual algorithm run by the agents is an extended version of the original BG, but extended in order to support a dynamic network.
We just provide the reader with the classic formulation of the BG since a more comprehensive discussion and details -also for the networked BG- are available in literature [8, 9, 7]. Here, we focus on the extension to support the structural dynamism.
The behavior of the agents is triggered by default at every time unit t. Between $t-1$ and t each agent eventually receives messages about received shipments and orders (i.e., demand). Then, at time t, every agent decides (i) how much to ship, (ii) how much to order and (iii) how to rewire:

i **how much to ship**: according to the demand coming from $DOWN_{Stream}$ and to any backorder that is still lying, an agent ships as many items as it can from its inventory, trying to address current and previous demand. Any order (demand) left unsatisfied is stored as backorder.

$$supply_t = min[Stock_t, Demand_t + Backorder_t]$$

ii **how much to order**: the decision is taken according to the expected demand for the next time unit and and to the adjustments performed on the stock and pipeline:

$$
\begin{aligned}
order_t &= Max[0, expDemand_t + Inventory_{adj} + Pipeline_{adj}] \\
expDemand_t &= \theta \cdot Demand_t + (1-\theta) \cdot expDemand_{t-1}
\end{aligned}
$$

where, $Inventory_{adj} = \alpha_S \cdot (Inventory_{desired} - Inventory_t)$ and $Pipeline_{adj} = \alpha_{SL} \cdot (Pipeline_{desired} - Pipeline_t)$ with $\alpha_S = 0.5$ and $\alpha_{SL} = 0.25$ representing the fraction of the gap to be closed among desired and actual levels [10]. Finally, $\theta = 1$ is a demand waiting

factor[2].

iii **how to rewire**: each agent keeps track of how many orders have been placed UP_{Stream} through each own neighbor. If no orders has been placed on neighbor B for k consecutive time units, the link with node B is removed. Since the number of orders placed on each neighbor is proportional to the neighbor's backorder queue, removing a link is tightly connected to the notion of the satisfaction of the relation between the parties. In other words, the longer the backorder queue, the longer the time to wait for a customer.

New links can also be created in the following manner. At regular intervals, each node A can pick at random an agent B from the set of agents belonging to their next UP_{Stream} group. For example, a retailer agent can pick any of the wholesaler agents. A link with B is created if its backorder queue length is shorter than the average of the backorder length of A's UP_{Stream} current neighbors.

5 Experimental scenario

5.1 Experiment setup

As a first scenario, we consider a synthetic SCN as in [9]. Five distinct stages are involved: factory, distributor, wholesaler, retailer and customer. The network looks like a 5 by 5 matrix as shown in Figure 2. The graph topology is fully connected; in other words, each agent is linked to every agent at the next and previous levels. Factory and customer agents, being respectively at the top and at the bottom of the structure are the exception. Factory agent are just connected with the previous level agents (distributors) and, vice-versa, customer agents are connected with the next level agents (retailers).

This setup is very similar to the one described in [9], but our topology is dynamic as agents can wire and unwire the connection with their neighbors according to their needs.

Experiments start by having the system in an equilibrium state. All agents have 12 elements in their inventory and in the pipeline, which correspond to their desired levels. The interval k is set to 3 time units. This is the consecutive time required to remove a link in case of an absent flow of orders UP_{Stream} between agents A and B.

The orders received equals to the order sent and customer demand is always satisfied in this manner. The agents behavior is totally deterministic and there is no random behavior. The system would stay in equilibrium forever if demand never changes.

We enable the agent ability to 'see' the orders in the pipeline at one stage beyond their neighborhood. This extra information would help to predict demand.

[2] $\theta = 1$ actually makes the current demand the only reference for the expected demand.

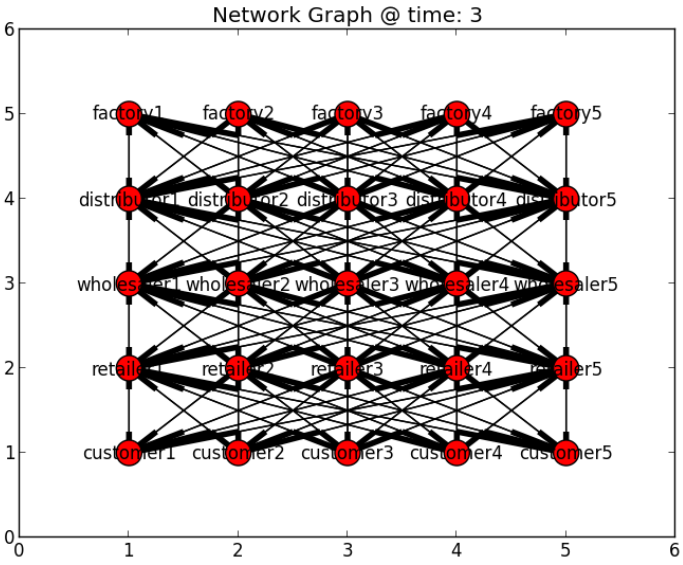
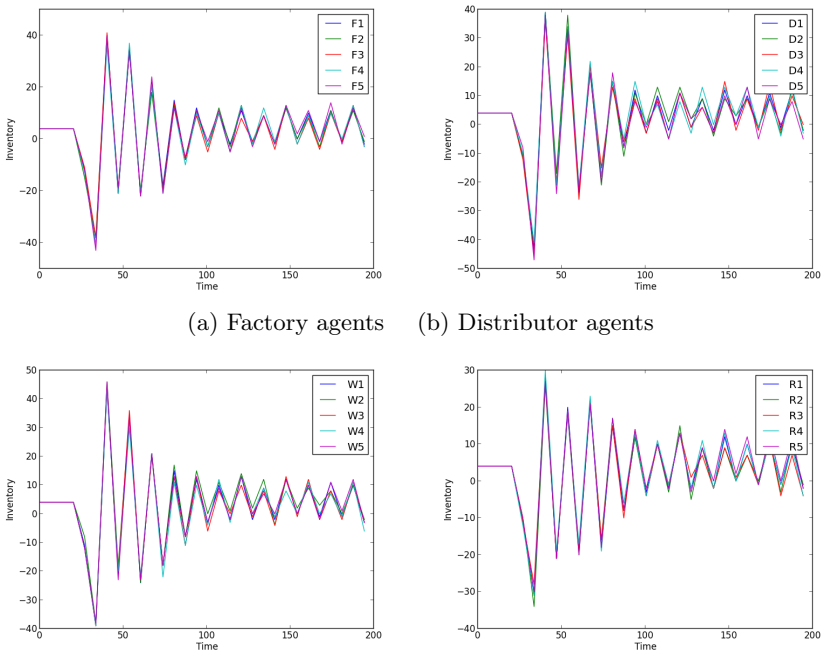


Fig. 2. The 5x5 supply chain network for the test scenario. Five distinct stages are present and each one is populated with five distinct agent. The topology is fully connected: each agent is connect with every all agents at the next and previous level. Customer and factory agents are the exception.

5.2 Experiments

The simulator triggers a change in customer demand at time 4. All customers ramp up (double) their demand from 4 to 8 items and the demand stays constant until the end of the simulation. The standard BG would start a chaotic behavior that would likely last forever. However, Figure 3 shows that by enabling the extended visibility on orders the system can sustain the ramp up of the orders. The visibility is enlarged by just one stage or hop in the network.
Each subfigure respectively shows the inventory level regarding factory, distributor, wholesaler and retailer agents. A distinct plot is shown for every agent.
The lowest level - the retailer's level - is the less affected one, but the effect of amplification is much more evident in the others sub-figures. After about 80 time units, the oscillations are much smoothed and the inventory level is close the desired one (12). This is quite an achievement considering the minimal information required and the imperfect allocation due to the parameter σ_c.
Figure 4 shows the topology after enabling the ability to remove links. The snapshot is taken at the end of the simulation. A few nodes has been removed from the topology since all their links gone and they were



(a) Factory agents (b) Distributor agents

(c) Wholesaler agents (d) Retailer agents

Fig. 3. Inventory level for each type of agent (e.g., factory, distributor, wholesaler, retailer). Every agent is represented by a single plot in each sub-figure.

unable to restore them. This is due to the small size of the network: the opportunities are few, because the customer demand is constant after time 4.
In addition, our current algorithm is designed to restore each own agent UP_{Stream} links, but an agent cannot manage its $DOWN_{Stream}$ links and it essentially relies on their $DOWN_{Stream}$ neighbors.
Periods with high demand would help to generate opportunities to accommodate a newcomer or would increase the chance for weakly linked node to acquire more valuable collaborations over time (e.g., find a niche).

6 Conclusions an future work

We have developed a supply chain model on top of our simulation framework ACP. We tested the model on a network supply chain exploring the value of spreading the information through the network. An important aspect that arises from this study is that a minimum information share is sufficient to smooth the chaotic fluctuations that characterize supply chains behavior [1, 3]. This suggests to push further effort on IT

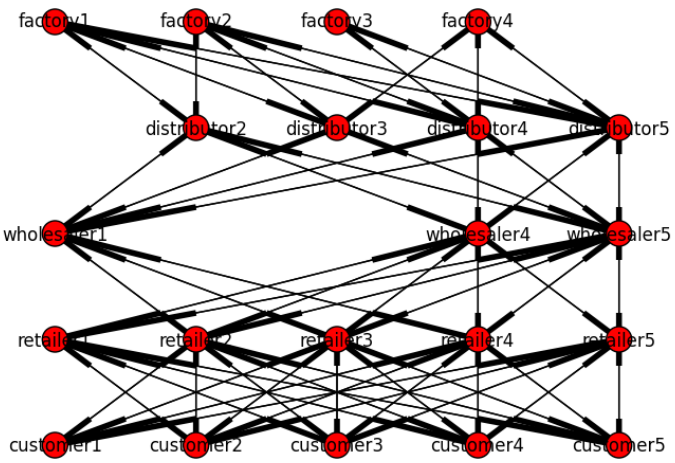


Fig. 4. Dynamic network. Agents have the ability to wire and unwire links among partners according to the feedback of the collaboration. A few nodes disappeared from the graph since all their links have been removed.

infrastructures in order to interconnect supply chains and consider them as organisms.

The results we obtained are similar to previous one obtained by other groups [9]. Essentially we consider it a proof of the validity of our model.

In or framework, dynamism is pushed to the limits since network evolution is involved and it is designed as a function proportional to the success of the dyadic relation over each link. This mechanism can pave the road to understand and predict the structural evolution of supply chains.

The next step for this preliminary work is to apply the model to a real world use case. In particular, we are focusing on food supply chain and using the data available online [11] in order to dissect the evolving structural aspect of supply networks.

References

1. Lee, H.L., Padmanabhan, V., Whang, S.: Information Distortion in a Supply Chain: The Bullwhip Effect. Management Science **43**(4) (1997) 546–558

2. Epstein, J.M., Axtell, R.: Growing artificial societies: social science from the bottom up. The Brookings Institution, Washington, DC, USA (1996)
3. J., S.: Business dynamics: Systems thinking and modeling for a complex world. McGraw-Hill (2000)
4. VV.AA.: The demand-driven supply chain: a holistic approach
5. Jesi, G.P.: ACP - Agents and Complexity in Python (October 2012)
6. Jesi, G.P.: Merging Event-Based and Agent-Based Simulation: The ACP Framework. Technical report, Social Science Research Network (SSRN) (2012)
7. Sterman, J.D.: Teaching takes off: Flight simulators for management education. OR/MS Today (October 1992) 40 –44
8. ArgonneNational Laboratory: Agent 2003 - Conference on Challanges in Social Simulation, Chicago, Illinois, ArgonneNational Laboratory (October 2-4 2003)
9. North, M.J., Macal, C.M.: Managing Businness Complexity - Discovering Strategic Solutions with Agent-based Modeling and Simulation. Oxford University Press (2007)
10. Sterman, J.D.: Expectation Formation in Behavioral Simulation Models. Behavioral Science **32** (1987) 190–211
11. Vasileiou, K., Morris, J.: The sustainability of the supply chain for fresh potatoes in britain. Supply Chain Management: an International Journal **11**(4) (2006) 317–327