Supply Chain Network Optimization based on Collective Intelligence and Agent Technologies

Leonid Sheremetov[*,1,2] and Luis Rocha-Mier[3]

[1]Centro de Investigación en Computación

Instituto Politécnico Nacional (IPN)

Av. Juan de Dios Batíz s/n casi esq. Miguel Othón de Mendizabal, Unidad Profesional "Adolfo Lopez Mateos"

Col Nueva Industrial Vallejo, 07738, Mexico D.F.

Mexico

sher@cic.ipn.mx

[2]St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences SPIIRAS, 39, 14th Line, St.Petersburg, 199178, Russia

[3] SAS Institute

Department of Analytics Intelligence

Mexico, D.F. C.P. 06500

México

luisenrique.rocha@sas.com

Abstract.

The supply chain network optimization is a difficult problem to solve in the context of distributed (information across different members) and dynamic (changes in the structure and content of the information) environment with multidisciplinary decisions

[*] Corresponding author, Centro de Investigación en Computación, Instituto Politécnico Nacional,
Av. Juan de Dios Batíz s/n casi esq. Miguel Othón de Mendizabal, Unidad Profesional "Adolfo Lopez Mateos" Edificio CIC.
Col Nueva Industrial Vallejo, 07738, Mexico D.F., Mexico.
Tel. (+52) 55-5729-6000 ext. 56576, Fax.: (+52) 55-5586-2936, sher@cic.ipn.mx

(employees' decision making at different levels). In this research work, an approach to the dynamic optimization of local decisions to assure global optimization in supply chain performance is developed within the frameworks of a Collective Intelligence (COIN) and Multi-Agent Systems (MAS). As a COIN, we mean a large multi-agent system where there is no centralized control and communication, but also, there is a global task to complete: the global supply chain network optimization. The proposed framework is focused on the interactions at local and global levels among the agents in order to improve the overall supply chain business process behavior. Besides, learning consists of adapting the local behavior of each agent (micro-learning) with the aim of optimizing a given global behavior (macro-learning). Reinforcement learning algorithms are used at the local level, while generalization of the Q-neural algorithm is used to optimize the global behavior. The framework is implemented over JADE agent platform. The experimental results demonstrate that this problem is a good experimental field for the investigation and application of the COIN theory.

Keywords: Supply chain network, multiagent system, optimization, collective intelligence, simulation.

Supply Chain Network Optimization based on Collective Intelligence and Agent Technologies

Leonid Sheremetov and Luis Rocha-Mier

1. INTRODUCTION

The ability to manage the complete supply chain network (SCN) and to optimize decisions is increasingly being recognized as a crucial competitive factor [17]. In order to make good decisions within a SCN, a manufacturer needs to coordinate local activities with those of upstream suppliers and downstream customers under uncertainty and imprecision in a very dynamic environment. Other entities in the chain are faced with a similar problem. Unfortunately, as the scope of supply chain management is extended, the underlying optimization problems grow dramatically in size. On the other hand, the availability of real-time status information is creating a need for 're-optimization' models and methods that can quickly repair a plan in response to changes in input data. This is particularly important for operational problems, where the solution to the optimization problem specifies a short-term assignment or schedule of resource use. As a consequence, logistics gets a new focus on optimization of the production process in a very dynamic environment [27].

Though there are many solutions and techniques for local optimization (*e.g.* planning and scheduling systems, inventory management systems, market trading optimization systems, etc.), usually these algorithms spend a lot of time finding the most appropriate solution and their decisions do not assure the overall business optimization at the global level because of the conflicts between the local goals of the different entities [13, 14]. Also, most of these approaches reflect today's supply chains model, which is essentially static, relying on long-term relationships among key trading partners. Recently several new optimization paradigms and approaches have been proposed. These new paradigms include both analytical methods (based on semidefinite optimization and computational differentiation) and simulation based optimization [5, 6, 7]. Some of the underlying mathematical issues involve development of fast algorithms to approximate the solutions to very large-scale problems, as well as methods for combining models of different types to form hybrid models. The issue of real-time updates to solutions has been addressed for a few specific applications, but no underlying theory has been developed. The size and difficulty of these problems make them impossible to solve with traditional optimization methods, and require the development of specialized combinatorial optimization techniques.

More flexible and dynamic practices offer the prospect of better matches between suppliers and customers permitting them react to a changing environment by adapting their operations [30]. This requires an approach, which can be flexible enough to accommodate imprecise linguistic data as well as precise numerical data, and which yields solutions that will provide compromise among different parties' objectives. To this end, there are few research works on theoretical treatment and different soft computing, machine learning and agent techniques developed for dynamic supply chain modeling and optimization in vague and uncertain environments [10, 11, 18, 24, 25]. In this paper, the problem of dynamic optimization of the Supply Chain Network (SCN) is addressed within the context of the COllective Intelligence (COIN) theory as an extension of Dynamic Programming models [29] and the adaptation of the Q-neural algorithm [21]. Within this framework, a SCN is a large Multi-agent System (MAS) where:

- One of the main objectives is the decentralization of the control and communication.
- Each agent of the SCN is represented as an agent with autonomous behavior and a local utility function.
- Agents model different SCN entities, the exchange of message and commodity objects among these agents emulates the information and material flows.
- The learning process consists of adapting the local behavior of each agent with the aim of optimizing a given global SCN behavior.
- The agents execute Reinforcement Learning (RL) algorithms at the local level while generalization of Q-neural algorithm is used to optimize the global behavior.

The rest of the paper is organized as follows. First, we describe the basics of Reinforcement Learning Theory as a theoretical background of the approach. In Section 3, a COIN Framework for the SCN optimization is presented. A special emphasis is made on the impact of the dynamic logistics decisions on the overall optimization of the system's performance and the adaptability against disturbances. Several algorithms composing the Q-neural optimization algorithm are described. In Section 4, we discuss an approach to dynamic SCN simulation within the COIN framework enabling SCN modeling as a dynamic, flexible and agile system of autonomous agents. A simulation test-bed for the evaluation of such types of multi-agent control architectures for SCN integrating discrete-event simulation facilities is implemented over JADE agent platform (AP). In Section 5, a case study is described along with the analysis of performance results of the simulation experiments of dynamic optimization of the SCN, followed by conclusions.

2. BASICS OF COLLECTIVE INTELLIGENCE THEORY

The Dynamic Programming theory forms the base of the RL methods used within the proposed approach [3, 4]. *Dynamic Programming (DP)* is a technique which addresses the problems which arise in situations where decisions are taken by steps, and the result of each decision is partially foreseeable before the remaining decisions are taken. An important aspect to consider in DP is the fact that the decisions cannot be made separately [22]. For example, within the SCN the desire to obtain a reduced cost in the present must be balanced against the desire to induce low costs in the future. This constitutes a Credit Assignment Problem since one must give credit or culpability to each decision. For an optimal planning, it is necessary to have an effective compromise between the immediate costs and the future costs. More precisely, Dynamic Programming focuses on the question: How can a system learn to sacrifice its short-term performance to improve its long-term performance? To answer this, DP relies on the application of the *Bellman's principle of optimality* defined as follows: *An optimal strategy p* is such as, whatever the initial state x(0) = i and the initial decision* $a_{i,k} \in A_i$*, the remaining decisions must constitute an optimal sub-strategy, with regard to the state resulting from the first decision.* In other words, an optimal strategy π* can be constituted only of optimal sub-strategies:

$$\{\mu_{x(0)}(0)^*, \mu_{x(1)}(1)^*, \ldots, \mu_{x(K)}(K)^*\}$$

where *K* is the horizon in time (or steps number). The Bellman equation presented below permits finding the optimal value function to solve the Markov Decision process [20]:

$$V_i^* = \min_{\mu i} \left(c_{(i,\mu_i)} + \sum_{j=0}^{N} p_{(i,a_{i,k},j)}\ V_1^*(j) | i = x(0), j = x(1) \right) \quad \text{for all } i \in \mathbb{X} \quad , \qquad (1)$$

where $c_{(i,\mu_i)} = \sum_{j=1}^{N} p_{(i,\mu_i,j)} \cdot g_{(i,\mu_i,j)}$.

p is the transition probability from the state *i* to the state *j*, when the action $a_{i,k}$ is completed.

$g_{(i,m,j)}$ is the cost due to the state transition from *i to j*.

V_i^* is the optimal value function such as:

$$V_i^* = \min_{\pi} V_i^{\pi}, \quad \text{for each state } i \in \mathbb{X}$$

In other words, the global optimization of the objective function is replaced by a sequential optimization that consists in optimizing each stage of decision (or period), one after the other, but by taking into account the former decisions which were made previously and the remaining decisions [9].

2.1. REINFORCEMENT LEARNING

Reinforcement Learning (RL) extends the ideas of the DP to treat more complete and ambitious goals of Artificial Intelligence (AI). Reinforcement Learning, instead of being like DP based only on the resolution of the problems of optimal control, is an aggregate of ideas from psychology, statistics, cognitive science and computer science [26]. To compute the optimal strategy, DP assumes perfect knowledge of the environment model (e.g. transition probabilities between the states of the environment and the costs (rewards/punishments) which the agent receives from this environment). The first question addressed by the DP was how to compute the optimal strategy with the minimum data-processing computation, by supposing that the environment can be perfectly simulated, without the need for direct interaction with it. The new trend in the methods of RL is the assumption of a limited (or even the absence of) knowledge about the environment model at the beginning and the prediction of rewards/punishments.

Moreover, instead of moving in a mental model internal space, the agent must act in the real world and observe the consequences of its actions. In this case, we are interested in the number of real world actions the agent must carry out to move towards an optimal strategy, rather than with the number of algorithmic iterations necessary to find this optimal strategy. The Bellman´s equation makes it possible to define an optimal strategy [16]. The systems which learn while interacting with a real environment and by observing their results are called online systems. On the other hand, the systems which learn separately, without interacting with a real environment and with an internal model of this environment are called off-line systems. In this work we are developing the SCN model as an on-line system.

Reinforcement Learning answers the question: *how to make the mapping between the states (of an environment whose model is not known entirely) and the actions of an agent (which interacts with this environment online) so as to maximize/minimize a numerical signal of reward/punishment?* In other words, within the context of the SCN this permits looking for the optimality of local decisions under the constraints of the optimal behavior of the whole SCN. The fact of learning by trial and error, and of having a delayed reward/punishment which will affect the future behavior of the agent are the two most important characteristics which differentiate it from other types of learning methods. A dilemma which arises in the field of RL and not necessarily with other types of learning is the trade-off which exists between the exploitation phase and the exploration phase. The agent must exploit the knowledge obtained until now to select the actions which brought it a high reward. But, at the same time, it must explore all the possible actions in its current state, in order to select an action which can bring it a higher reward than the actions carried out in the past. This dilemma has been studied by several mathematicians [26].

2.2 Q-learning ALGORITHM

One of the mo st important advances in the field of RL was the development of Q-learning [28], an algorithm which follows an off-line strategy [26]. In this case, the value-action function learned, *Q*, approximates the optimal value-action function *Q** in a way independent of the strategy followed. In state *x(t)*, if the Q-values represent the environment model in an exact way, the best action will be that which has the most/less important value (according to the case) among all possible actions $a_{i,k} \in A_i$.

The Q-values are learned by using an update rule which uses a reward *r(t+1)* calculated by the environment and a function of Q-values of the reachable states by taking the action $a_{x(t)}$ in state *x(t)*. The update rule of Q-learning is defined by:

$$Q_{(x(t),a_{x(t)})}(t+1) = Q_{(x(t),a_{x(t)})}(t) + \alpha\left[r(t+1) + \gamma \min_{a_{x(t+1)}} Q_{(x(t+1),a_{x(t+1)})}(t+1) - Q_{(x(t),a_{x(t)})}(t)\right] \quad (2)$$

where:

- α is learning rate, γ is reduction rate.
- The Q-values $Q_{(x(t),a_{x(t)})}$ give an estimation of the environment. The way in which the Q-values are updated can be considered as one of the most important problems to solve in our framework.
- The reinforcement for the performed action is represented by $r(t+1)$. This function of reinforcement represents the partial time of product production composed of: a) transition time, b) waiting time, and c) operation time.

The Algorithm 1 shows the Q-learning algorithm [28]:

```
Initialize:
  The Q-values $Q_{(x_i,a_{x_i})}(0)$ in an arbitrary way  for all the states $x_i \in \mathbb{X}$
   and all actions $a_{x_i} \in \mathbb{A}_{x_i}$
  The strategy $\pi($ e.g. $\epsilon$-greedy)
  $t = 0$
  $\alpha, \gamma$
repeat
  (for every episody)
    Initialize $t = 0$ and the state $x(t)$
    repeat
      (for every step $t$ of the episody)
        Choose an action $a_{x(t)}$ from the state $x(t)$ and using the strategy derived from $Q$
        $a_{x(t)} = \mu^{\pi}_{x(t)}$
        Take the action $a_{x(t)}$, observe the reward $r(t+1)$ and the next state $x(t+1)$
        $Q_{(x(t),a_{x(t)})}(t+1) = Q_{(x(t),a_{x(t)})}(t)+$
        $\alpha\left[r(t+1) + \gamma \min_{a_{x(t+1)}} Q_{(x(t+1),a_{x(t+1)})}(t+1) - Q_{(x(t),a_{x(t)})}(t)\right]$
        $t = t+1;$
        $x(t) = x(t+1)$
    until $x(t)$ is the terminal state
until
```

This RL method makes it possible to solve problems of learning for only one agent. Nevertheless, when several agents act in a common environment, these methods are not very effective. There is a lack of learning mechanisms for systems comprising a large number of agents. In our research work, these problems are addressed by using the theory of the COllective INtelligences presented in the next section. We think that the question of collective learning will become one of the principal questions to solve in years to come.

3. COIN Framework for the SCM Problem

The term "supply chain" has been used since the 1980s to describe the whole spectrum of operations in almost every manufacturing industry; from purchasing of raw material, through transformation production processes, to distribution of the finished inventory to customers. As the complexity increases a supply chain is well depicted as a network of suppliers, manufacturers and customers (Fig. 1).

Insert Fig. 1 here.

This work proposes a model of the SCN in the framework of the COIN theory. As defined in [8], SCN members are believed to have common functions. These common functions are handling of incoming and outgoing flows, flow transformation and control. Materials in the SCN are represented as objects forming part of the environment. Therefore, every agent can change or influence these environment objects. The details of the objects are stored as attributes. In our approach, an agent can represent any member of the SCN. Each agent has a local utility function and handles a Q-table, which contains perceived information about the environment (both environment objects and neighbor agents). A generic scheme of an agent is depicted in fig. 2. This generic scheme can be applied both to the members of the SCN and to the components of each member depending on the necessary level of decomposition. A network of agents represents the entire SCN. This representation enables dynamic SCN simulation within the COIN framework.

Insert Fig. 2 here

According to figures 1 and 2, we define the following elements of the COIN framework:

- Set of n distribution warehouse agents (DWA) – the head of the chain- that has the knowledge on final products orders PO: $FP = \{FP_1,...,FP_n\}$.
- Set of m producer agents (PA) – manufacturers, assemblers and component suppliers – intermediate members of the chain having both input and output flows: $M = \{M_1, M_2,..., M_m\}$.
- Set of k supplier agents (SA) – the tail of the chain: $S = \{S_1, S_2, \ldots, S_k\}$.
- Set of s operations executed by agents $i : OP_i = \{O_1,..., O_S\}$.
- Vector of non-negative values of r features for each operation $O_i : \overline{V_i} =< v_1^i,..., v_r^i >, e.g. v_1^i$ = average time[2].
- Set of s objects corresponding to a type of raw material: $MP = \{MP_1,..., MP_S\}$.
- Set of n objects corresponding to a type of final product: $P = \{P_1,..., P_n\}$.
- Vector of non-negative values of r features for each product $P_i : \overline{PV_i} =< pv_1^i,..., pv_r^i >$, e.g. pv_1^i - product priority.

In this work, each agent has the following features:

- The set of environment states $X = \{x_1, x_2, x_3,...\}$. Knowledge (usually incomplete) about other agents is considered to be part of the environment state. For example, in some cases a manufacturer agent might make decisions without knowledge that a supplier has frequently defaulted on due dates.

- The capacity of agent to act is represented as a set of actions: $A_i = \{a_1, a_2, ..., a_k\}$. Note that for PAs (manufacturer agents) this set is equal to the *OP* set.
- The relationships between the agents in the SCN are defined by: $R = \{r_1, r_2, r_3, ...\}$. Agents known to the current agent form the list of his neighbors: $N = \{n_1, n_2, n_3, ...\}$. In the case of the linear model, only agents from the nearest tier are included in this list. For each neighbor agent the following parameters are considered: a) its relationship to the current agent (customer, supplier), b) the nature of the agreement that governs the interaction (production guarantees) and c) the inter-agent information access rights (the agent's local state to be considered during the decision-making process).
- The priorities of every agent are represented by: $Q = \{q_1, q_2, q_3, ...\}$. These priorities can help in sequencing incoming messages for processing.
- The local utility function (LUF) is represented as the Q-learning equation (2).
- The set of control elements: $C = \{c_1, c_2, c_3, ...\}$. A control element is invoked when there is a decision to be made while processing a message. For example, in order to determine the next destination in the transport of materials, a routing-control algorithm would be utilized.
- Every agent has a message handler that is responsible for sending and receiving different messages to facilitate communication among the agents.

To address the SCN optimization problem, the adaptation of the Q-neural algorithm [21] is proposed and described. The behavior of the Q-neural was inspired by the Q-routing algorithm operation [15], the theory of COIN, including the algorithms based on the behavior of the colonies of ants. The learning is done at two levels: initially, at the agent's level locally updating the Q-values by using a RL rule, then, globally at system level by the utility function's adjustment. The control messages allow updating knowledge of the SCN entities by updating the Q-values, which are approximated by using a function approximator (look-up table, neural network, etc.). In Q-neural, there are 5 types of control messages:

- *An 'environment-message' (flag_ret=1)* generated by an intermediate PA after the reception of a raw material if the interval of time ω has already passed.
- *An 'ant-message' (flag_ret=2)* generated by the DWA according to the interval of time *w_ants* when a final product arrives at the warehouse.
- *An 'update-message' (flag_ret=3)* generated in the planning phase every *e__update* seconds to ask the neighboring PA their estimates about the operations of the product.

[2] The features vary from one agent to another.

- *An 'update-back-message' (flag_ret=4)* generated after the reception of an update-message in order to accelerate knowledge of the environment.
- *A 'punishment-message' (flag_ret=5)* used to punish an PA using a congested resource.

The Q-neural algorithm includes 3 parts: planning, ant-message and punishment algorithms described below.

3.1. Planning as Exploration

Exploration can involve significant loss of time! In Q-neural, a mechanism of planning (within the meaning of this term in RL) was developed at the local level of each agent. This mechanism consists of sending an update-message every e_update seconds. This update-message will ask for the Q-values estimates of all the products, which are known at that moment by the neighbors.

Algorithm 2: Planning Algorithm in "Q-neural"

```
Every $e\_update$
    Send an update-message (flag_ret = 3) to all the neighbors to ask their estimates of all the known products
if (an update-message is received) (flag_ret = 3)
    Send an update-back-message (flag_ret = 4) to the source agent of the update-message with the estimates $Q^{m}_{(x(t),a_{x(t)})}(t)$ of all the known products appearing at instant $t$
if (an update-back-message is received) (flag_ret = 4)
    Update the Q-value in the same way as is used for an environment-message (flag_ret = 1)
```

3.2. Ant-message Algorithm

When an environment-message arrives at the DWA, an ant is sent in return if the period of time *w_ants* has already passed. This ant exchanges the statistics obtained on its way and it allows the environment information communication among the agents. When it arrives at the storage of raw materials, it dies. The ant updates the Q-value of each PA through which the raw material passed before arriving at the DWA.

Algorithm 3: Ant-message algorithm in "Q-neural"

```
if (a product arrives at DWA)
then the DWA generates an ant-message if the time $w\_ant$ have passed
```

```
if (an agent $M_i$ receives the ant-message from the neighbor agent $M_y$ or DWA)
then
    Read the estimate $Q^{A}_{(x(t),a_{x(t)})}(t)$ from the header of the ant-message (flag_ret = 2)
    Consult the best estimate at present time $Q^{M}_{(x(t),a_{x(t)})}(t)$
    if $Q^{M}_{(x(t),a_{x(t)})}(t) > Q^{A}_{(x(t),a_{x(t)})}(t)$ and (a cycle is not detected)
    then Update the Q-value by using $Q^{M}_{(x(t),a_{x(t)})}(t)$
    else Do not update
```

3.3. Punishment Algorithm

In some cases, different agents from the same tier can have the same best estimate (prefer the same route). If they act in a greedy way, congestion occurs in the queue. To avoid congestions, an agent must sacrifice its individual utility and to use another route. In order to address this problem a punishment algorithm is developed forcing an agent who receives a punishment message to calculate the second best estimate.

Algorithm 4: Punishment Algorithm in "Q-neural"

```
if a punishment-message is received by $M_y$ from $M_z$
    Compute the second best estimate $Q^{M_y}_{(x(t),z')}(t)$ to arrive at the DWA by using a line that is not $l_{y,z}$
    Send a message to all the neighbor producer agents $M_{xi}$.
    Receive the second best estimate of every neighbor $M_{xi}$.
    Select the best estimate among all the estimates of the neighbors: $\arg\min Q^{M_{xi}}_{(x(t),y')}(t)$
if (the second estimate $Q^{M_y}_{(x(t),y')}(t)$ exists)
    if (the best estimate $\arg\min Q^{M_{xi}}_{(x(t),y')}(t)$ of the neighbors exists)
        if $((Q^{M_y}_{(x(t),z')}(t) < (\arg\min Q^{M_{xi}}_{(x(t),y')}(t))))$
        Punish the line $l_{y,z}$ : $Q^{M_y}_{(x(t),z)}$ =second estimate $Q^{M_y}_{(x(t),z')}(t) + \Delta$
        else
        Punish the line $l_{xi,y}$ : send a punishment-message
    else
```

```
        Punish the line $l_{y,z}$ : $Q^{M_y}_{(x(t),z)}$ =second estimate $Q^{M_y}_{(x(t),z')}(t)+\Delta$
else
    if (the second best estimate $\min Q^{M_{xi}}_{(x(t),y')}(t)$ of the neighbors exists)
    Punish the line $l_{xi,y}$ : $Q^{M_{xi}}_{(x(t),y)}(t)$ =second estimate $Q^{M_{xi}}_{(x(t),y')}(t)+\Delta$
    Send a punishment-message with the estimate
```

Finally, the Q-neural algorithm is defined as follows:

Algorithm 5: "Q-neural"

```
Initialize at $t=0$: All the Q-values $Q_{x(t),a_{x(t)}}$ with high values, the RL parameters: $a$, $g$, exploration, $w, w\_ants$
REPEAT
Update the instant $t$
if a material is received by the producer agent $M_i$
    Read the input vector $\bar{x}$ from the material header and environment variables
    Send the message to the agent $M_x$ where the material arrives with the value of the reinforcement function $r_{(t+1)}$ and the estimation
$Q^{m}_{(x(t),a_{x(t)})}(t)$
    Execute the operation $O'$ and choose the action $a_{\bar{x}(t)}=M'$ in function of the input vector $\bar{x}$ by using the strategy $e\_greedy$ derived
from $Q_{(x(t),a_{x(t)})}(t)$
    Send the material $a_{\bar{x}(t)}=M'$
    At the next time step, receive the message from the agent $M'$ with the value of the reinforcement function $r(t+1)$ and the estimation
     $Q_{x(t+1),a_{x(t+1)}}(t+1)$
    Apply the Q-learning update rule:
$Q_{(x(t),a_{x(t)})}(t+1)=Q_{(x(t),a_{x(t)})}(t)+a$
$\left[r(t+1)+g\min_{a_{x(t+1)}}Q_{(x(t+1),a_{x(t+1)})}(t+1)-Q_{(x(t),a_{x(t)})}(t)\right]$
REPEAT
    Algorithm of planning $e\_update$
    Algorithm of punishment
    Algorithm of Ants
```

4. Multiagent Framework for Supply Chain Modeling and Optimization

COIN optimization algorithm described above is implemented within a more general multiagent framework for supply chain modelling and optimisation – MASCS [23]. This framework represents complex dynamic interactions among supply chain members, accounts for demand uncertainty, validates and if necessary allows modification of configuration, optimization and coordination results. It can be used to improve decision-making within a wide range of problems in various supply chain scenarios. The relevant decisions can be classified into three categories [12]: strategic, operating, and control. Strategic decisions such as selecting the supply chain participant have long-term significance. Operating decisions refer to decisions about production to meet demand. Finally, control decisions are concerned with problems in execution. This can be classified as disruption management (like situations when a certain machine in the shop floor fails).

Though being more oriented on the solution of the second type problems, the framework can handle the other categories as well. In the case study we show how simulation results can be used for the SCN configuring based on the performance analysis. On the other hand, the algorithm of dynamic optimization suits well for just-in-time decision making.

Fig. 3 provides the schematic of the developed approach. At the stage of SC template configuration a user can specify three types of parameters: (i) define operational parameters and specifications, (ii) determine structural requirements to the SC, and finally (iii) define performance goals. Operational parameters include inventory control policies, production capacities, order lead-time, etc. For a selected type of a supply chain, the historical production data and BOM information can be obtained from the ERP system (at the moment the interface to the Excel based MRP is implemented [19]). All the products composing a demand are broken down to their component parts according to the BOM. Demand parameters for each component are obtained as a result of the forecasting based on estimation of the patterns from the historical data. We use perceptual forecasting module from the Fuzzy Toolbox Library developed by the authors for time series analysis [1].

Insert Fig. 3 here

From here the simulations using COIN model are executed iteratively, beginning with a supply chain structure (manufacturers, suppliers, customers, etc.) and a specific set of operational parameter settings (inventory levels, production capacities, lead-times, etc.) until a system configuration is achieved, providing a compromise between specified performance and management goals (such as "demand satisfaction level to be *high* and inventories to be *low*"). Since many possible combinations and lines of action are possible to improve the whole system, usually a considerable amount of time has to be spent trying to change the original system searching for a good design and balancing several conflicting objectives simultaneously. To avoid this tedious procedure fuzzy rule engine (described in the next section) is integrated with the simulation of the supply chain.

Actually, the simulator is a COIN-based multi-agent system where each node of the SCN (Fig. 1) is simulated by an agent. The operation of the system is simulated for a period of time and performance measures calculated. Observed performance is then compared with stated goals. If the supply chain objectives are not yet achieved, the system will request its fuzzy knowledge base for adjustments having its latest system performance measures on hand. During this dialog, fuzzy rules contained in the knowledge base will be activated to adjust parameters in the simulation model. This process is repeated until the system objectives are met to a high degree. Once simulation is over, the report with performance measures is generated and the data for each element of the SCN (likecomponents and product pivots or purchase orders) obtained as a result of simulation can be stores back to the ERP system.

4.1. The Architecture of the COIN Simulator

Usually the following SCM streams are considered: upstream (procurement), internal (manufacturing, assembly), and downstream (distribution). We simulate three types of participants. First, suppliers, which are in charge of supplying parts to other participants, but do not receive any. Suppliers are simulated in our model as warehouses generating raw material flows. We suppose that a sufficient amount of raw materials is available all the time for supplier tier, which are generated on demand. Second, producers (sometimes also called traders) are the intermediate participants in the supply chain (manufactures and assembly units) both placing orders with some participants (pertaining to the previous echelon) and delivering orders to other participants (pertaining to the next echelon). Buffers (inventories) are located before every element (operation); products are transported between all stations. There is an important difference between manufacturing and assembly elements: in the latter case, several buffers (one for each different component) should be located before the element. Without loosing the generality we consider for the case study only products composed of 2 components at most, so the assembly element will have 2 input buffers (Fig. 4). Each echelon of the chain has parallel processing units and limited inventory buffer capacity. Though in the case study we shall consider the linear SCN but as shown in [10], the model can be easily extended also to the non-linear case.

Insert Fig. 4 here

Each producer agent may be implemented at different level of detail. For example, a manufacturer can be modeled as a member of the SCN, with an approximate measure of factory-wide parameters such as production lead-time. On the other hand, it can be represented at a very detailed level, as a MAS itself, with agents representing the operation of each machine in the factory and the flow of raw material on the shop floor. For these purposes, each producer also has an Excel sheets associated with it containing BOM information, components and product pivots, purchase orders. Performance measures are also calculated based on the entries in the corresponding Excel sheets. It is important to mention that the simulation model permits considering jobs with different required times (just-in-time manufacturing).

Finally, customers (distributors or retailers) place orders and implement delivery processes instead of manufacturing.

The performance measures usually include cycle time, percent tardiness, inventory, cost performance, and resource utilization. The simulator has the following measures (cost performance measures are not considered at the moment):

- Order based performance measures are calculated based on the orders that have been delivered during any given period. The overall cycle time is calculated as the average time between the placing of an order by the customer and the delivery of that order by the producer (or trader) at the customer site. Each product has an estimated lead-time. If the order is delayed beyond its estimated delivery date, then the order is considered tardy.
- Delivery performance includes the average cycle time at each stage, the overall cycle time, and the percentage of orders that were tardy.
- For calculating the resource utilization, variables are used to keep track of the amount of time the resource was busy in any given period.
- Inventory performance measure is the average of the inventory at the beginning of the period and the inventory at the end of the period. The inventory performance measures include raw material, work in process, and finished goods inventory.

To test the validity of our approach, a simulation model for a simple five-echelon supply chain was created and tested on a capacity allocation scenario. The model was implemented in the agent-based parallel modelling and simulation environment over the JADE agent platform and Netlogo simulator for results visualization [2, 18]. CAPNET Expert System Shell was used as a rule engine.

4.2 Implementation of the Q-neural Algorithm within the Multiagent Framework

The above-described COIN model of the SCN has been implemented in order to test the performance of the developed algorithms for dynamic optimization of the SCN. For synchronization purposes utility Scheduler Agent (SCHA) was added to the multiagent system. Each customer places orders for any of the simulated products. Actually, customers are modeled by Dummy Distribution Warehouse Agents (DWA) which generate the orders dynamically using forecasting algorithms [1] for each customer to simulate this phase. Once the order is generated, it is placed with the corresponding producer agent (PA) of the internal stream of the SCN. Producer agents in turn decide which components to procure with the suppliers. During the order placement phase the best route is dynamically obtained considering the available capacity of the SCN members stored in the Q-tables. This route is followed further during the simulation of the production process.

The simulation begins when the SCHA broadcasts a *startUp* message to the whole group. This way, agents can perform internal initialization tasks mainly related to the tables (known as Q-tables) and variables to be used by the Q-algorithms. In the case of the DWA, it will load the *Technological Processes and Orders* lists that will be processed. It will invoke first the forecasting module in order to determine the production program. Once the program is generated, it replies to the SCHA with the initial list of orders to be released. We do not simulate explicitly the orders' due dates. Instead, the actual simulated date of the finished order is compared with the due date. Those orders which due date has expired are placed in the tardy orders list. Products inherit the priority of the orders which is related to the orders' due dates.

Material flow is initiated by the suppliers providing the raw materials according to these purchase orders. Then, producer agents take the raw materials from the warehouses to produce the products according to the BOM. Raw materials, intermediate products and final products are not physically present. The information concerning each of them is passed via message interchange between the SA and PA. A raw material becomes a final product after being processed in the SCN. Initially, this object is created with an operations vector that decreases at each step, until it gets empty and a final product is located at the corresponding warehouse.

The SCHA warns PA each time they must take some product from their internal queue and get it processed. However, this is only a notification; the actual processing will not take place until the SCHA informs a PA to do so. After the operation is completed, the agent is responsible for routing the product to the next tier where two events are triggered: a neighbor's request to add the intermediate product and a SCHA's request to add a new event for the corresponding agent to check its buffer for pending jobs. Also, there is a ping-like operation that ensures the selected agent is still alive. Otherwise, the second best alternative will be chosen and the corresponding PA will be notified to queue the product for further processing. In consequence, the Q-tables are changed since a new product has been set in the link between the current agent and its neighbor.

During the initialization stage, PAs search their local Directory Facilitator. They receive a list of partners from the next tier of the SCN Creating Neighbor Agents Table. Also, PA asks neighbors (located at the next tier) about their capabilities and available capacity. This information is used to initialize the Q-tables. Data back-propagation to the previous stage is achieved after a PA commits to queue a product. This mechanism helps ensuring that each agent has information to optimize routing for a product. Q-algorithms are embedded in PA's body. This results in information updating aimed in optimizing the decision making process on the best route selection for each stage.

It must be noticed that a DWA works partially as a PA for the following reason: a DWA also keeps a Q-table for making decisions on where to send the order after this is released. If the PA detects that there are no pending operations to accomplish for the processed product, it will notify the DWA. In turn, this agent will modify the stock numbers for such product.

For communication purposes, different types of events that the agents must be aware of are encoded in messages such as: “raw material release” (RELEASE_MP), “final product report” (INCR_STOCK) and “product leaves agent’s buffer” (DEQUEUE). Message structure between MA and SA is slightly different from conversations with the SCHA. This latter manages only events notifications from and to the group of agents for simulation purposes. PA and SA, on the other hand, implement action requests for operations (Fig. 5). There, PA1 requests PA2 to add a product to PA2’s queue. That is the way products travel from tier to tier. In the last conversation, PA3 requests PA4 and PA5 to inform their capabilities in order to update PA3’s Q-tables. Each PA responds to the query with the list of operations it is able to perform and associated processing time using FIPA-query interaction protocol. These operations are specified as “operations concept” using domain ontology.

Insert fig. 5 here

As shown in fig. 9, PAs are distributed along the JADE containers according to the tier they belong to. Also, container facilities work as a bridge for experimenting with distributed locations. There is an option to connect physically distant machines that resembles the common layout of a real SCN. In other words, the implementation we present is two-folded. It can be used as a test-bed for trying different configurations, and if required, it is intended to function as an implementation on a real network.

4.3. Fuzzy Rules

Relationships between performance measures (e.g., service level) and system parameters (e.g., number of processing units, inventory levels, elements’ capacities, and lead-times) in supply chains are usually understood and characterized in an imprecise, linguistic fashion; for example, "enhancing service by shortening lead-time". However, the precise relationships between system parameters and performance measures are really not known and, in fact, will change from one time to another depending on uncertain factors. While the relationships between system parameters and performance cannot be described by precise mathematical functions, fuzzy mathematics permits one to directly model imprecise relationships using linguistic variables.

Different rules define the logic of the SCN operation. For example, the make-to-inventory logic rules include:

- the products with the low cover (compared to the purchase order due-dates) are identified as the most priority products to be made,

- the products to make is that required to increase the inventory level to the maximum, plus sufficient to replenish the amount sold while the job was running,
- if the projected inventory level of the lowest cover product is above its minimum, then an idle period is scheduled.

Fuzzy approximator provides the theoretical basis for developing "if-then" rules such as:

IF the service level for customer C1 is "too low", AND the use of unit X capacity at echelon Y is "very high" THEN increase the capacity at echelon Y by a "medium" amount.

As shown in the rule example above, imprecise concepts such as *low* customer service and *medium* capacity are modeled as fuzzy sets. The membership function (MF) definition component of the Toolbox permits modellers to choose the shape of the membership function from a pool of commonly used parameterized families including triangular, trapezoidal, Gaussian, sigmoid, and S-shaped. Figure 6 illustrates a definition of the MF used to define echelon's capacity. In this case, capacity levels around 50% (of the demand) are regarded as definitely "medium" and thus have membership values at or close to 1. On the other hand capacity levels below 10% and above 90% are definitely not "medium" and thus have membership values of 0. Points in between have memberships which rise toward 1 the closer they are to 50%.

Once, the numerical values of the parameters are obtained during the simulation, the linguistic values used in the rules are generated using another component of the Toolbox Library according to the defined membership functions. The fuzzy rules are used to guide the operational parameter adjustment. In the case study described below, we identify the semi-optimal echelon's capacity level fo r different demands by adding or reducing the processing units.

Insert fig. 6 here

5. A CASE STUDY AND EXPERIMENTAL RESULTS

The description of the case study and simulation results are explained in more details below. The purpose of this section is twofold: (i) to show how locally optimal decisions made by the SCM members in a greedy way can influence global SCN parameters, and (ii) show a typical example of how the MASCS could be used to make managerial decisions integrating both macro- and micro-level optimization. For the purposes of this study, we consider macro-level problems to be focused on minimization of the total supply chain costs (inventory level optimization: coordination of stock levels, and capacity level optimization: coordination of production levels through the SCN). On the other hand, micro level optimization is concerned with the goals of supply chain members under constraints set by macro-level solution (capacity levels and production priorities). More particularly, in the example shown below we persuaded the goal to apply the proposed framework to obtain the semi-optimal echelon's capacity levels using parameters adjustment with fuzzy rule engine.

In the case study we try to capture some characteristics of a virtual SCN such as: (i) a high complexity of SCN, (ii) the need for reciprocal means for suppliers to commit to orders on line with a service that's integrated with manufacturing planning systems, (iii) high sensitivity to the inventory levels, and (iv) the need to automatically determine the best manufacturing facilities for specific orders simultaneously taking into account all supply chain data and all constraints in every plant. For illustrative purposes, we consider a simple example which consists of 3 raw materials (A, B, C) and 2 final products (Table 1). The technological process is the following (Table 2): each unit of Product 1 consists of one unit of raw material A (after the processing operation O11) and two units of raw material B (after the processing operation O12). These units are assembled (operation O22) and then processed by O33. Each unit of Product 2 consists of the processing operations O11, O23 and O33 over C raw material. Suppliers S1, S2 and S3 supply raw materials A, B and C respectively. Finally we simulate three customers (agents DWA1-DWA3) generating their orders dynamically through the first day of the week. In a basic configuration, each operation has time duration according to the Table 3. Original layout is shown in fig. 7.

Table 1. Product range

Code	Description	Forecast Products per Week
P1	$((AO_{11}+ BO_{12})O_{22})O_{33}$	600
P2	$((CO_{11})O_{23})O_{33}$	1000

Table 2. BOM: 4 products P1 – P4.

Product	Component	Supplier	Qty
P1	A	S1	600
P1	B	S2	1200
P2	C	S3	1000

Table 3. Operation duration.

Agent	Operation	Duration	Agent	Operation	Duration	Agent	Operation	Duration
A11	O11	3	A21	O22	1	A31	O33	1
A12	O12	1	A22	O23	5	A32	O33	9
A13	O11	1	A23	O23	2	A33	O33	6

Insert fig. 7 here

Experimental results are shown in the product Pivot (Table 4). Figures 8 and 9 show the overall cycle, average time between the placing of an order by a customer and the delivery of the product to this one, and average inventory levels respectively obtained using different algorithms. First we used the traditional Q-routing algorithm. The agents based in Q-routing, make their decisions in a greedy way to optimize their local utility functions. This models the situation if we were managing individual sites without knowing what the others were doing. This conduces to high inventory levels, and a decrement of the global utility function is obtained as a result of this greedy behavior. As shown, for the original SCN configuration (fig. 7) Q-neural algorithm (line 1) shows better performance compared to Q-routing (line 2) due to the fact that agents make their decisions taking into account the global SC performance, not only that of the local level. As a result, the performance of the scenario is improved thanks to the adaptation to the changes in the SC environment.

Table 4. Product Pivot

Prod/Run	Product	Total	Run
P1/1	P1	599	1
P1/2		600	2
P2/1	P2	870	1
P2/2		999	2

Insert fig. 8 here

Insert fig. 9 here

The second type of experiments is based on the results obtained during the simulation run using Q-neural. According to the analysis of the obtained results during the first run, we can deduce that the model of the Fig. 6 (denoted as an original structure) is not optimal because of the long time duration for the operation O23 which results in high utilization of the agents PA22 and PA23. By applying the corresponding fuzzy rules, developed within the framework of this research work, the structure of the simulation model is iteratively adjusted for the following simulation runs until an acceptable performance is obtained. As shown in Fig. 10, an important adjustment in the SC structure was the additional agent PA24. This adjustment avoids the bottleneck detected in the original structure (composed of two agents PA22 and PA23 for the operation O23).

As we also can see from fig. 9, the model rations the inventory across the products so as to even out the product inventory cover as much as possible. The reduction of the amount of on-hand inventory required as a buffer increases efficiency. Inventory levels in the example were reduced by 10% -20%.

Insert fig. 10 here

Finally, the scalability tests were conducted. For the case study, production of 1600 products took about half an hour of simulation running in the server mode, almost the same for the SCN consisting of 25 agents. Later, we configured 5 echelons consisting of 10 entities each. The simulation took about 1 hour. This time can be considerably reduced in a distributed mode, which is natural for agent technology.

6. CONCLUSIONS

Today's challenge is to optimize the overall business performance of the modern enterprise. In general, the limitations of traditional approaches to solving the problem of dynamic global optimization of the SCN are due to the fact that these models do not correspond to the reality because of incomplete information, complex dynamic interactions between the elements, or the need for centralization of control and information. Most of heuristic techniques on the other hand, do not guarantee the overall system optimization.

In this paper, the problem of dynamic optimization of the SCN is addressed within the framework of the COIN theory. In order to optimize the global behavior of SCN, learning process using RL algorithms to adapt agent’s local behavior is used. Though for the case study only linear model of the SCN was used, non-linear schema can also be implemented without any changes to the model. The only thing to be done to model a network is to redefine the concept of a ‘neighbor’ enabling connections of the entities pertaining to the different tiers.

The model was implemented in the agent-based parallel modeling and simulation environment over the JADE platform. Being the agglutinating centre of the enterprise information infrastructure, an AP also serves as an experimental test-bed for the implementation of the models developed in this paper. By means of this, we can easily implement the algorithms tested in the simulated environment into the real-world applications.

This conceptualization permits to handle the SCM problem within the following context:

- The environment is of a distributed and dynamic nature.
- The model of the behavior of the environment at the beginning is unknown.
- The individual behavior of each agent affects the total behavior of the system. One of the problems is to find out, which part of the system affects the total behavior. Another problem is to know the degree of responsibility for each agent also known as 'Credit Assignment Problem.'
- The number of states and variables is very significant.
- The entities must adapt their decisions online.

In the on-going experiments, the tested control systems will have varying production volumes (to model the production system with looser/tighter schedules) and disturbance frequencies, so that the impact of the SCM and sequencing decisions in various manufacturing environments can be evaluated. We also pretend to compare our algorithms with other classic optimization methods using the developed multi-agent test-bed.

Though we do not have yet the results of these experiments, we conclude that the SCM problem is well suited for the application of the COIN theory. In addition, the adaptive Q-neural algorithm provides better throughput and reliability than other algorithms. In future work, an adapted model of the CMAC Neural Network will be used for the Q-values approximation. More complicated punishment algorithm will be developed to adjust the local utility functions.

ACKNOWLEDGMENT

The authors would like to thank Ana Luisa Hernandez, Juan Guerra y Jorge Martinez for their help in implementing of the described framework and providing experiments.

REFERENCES

[1] I. Batyrshin and L. Sheremetov, Perception Based Time Series Data Mining with MAP Transform. MICAI 2005: Advances in Artificial Intelligence. In A. Gelbukh, A. Albornoz, H. Terashima-Marín, eds., Lecture Notes in Computer Science, Springer Verlag, 3789: 514 – 523, 2005.

[2] F. Bellifemine, A. Poggi, and G. Rimassa, JADE - A FIPA-compliant Agent Framework. In *Proc. PAAM’99*, London, UK, 1999, pp. 97-108.

[3] R. Bellman, On a routing problem. Quarterly of Applied Mathematics, 1958.

[4] D. P. Bertsekas and J. N. Tsitsiklis, Neuro-Dynamic Programming. Athena Scientific, Belmont, Massachusetts, 1996.

[5] D. Bertsimas, A.Thiele: A Robust Optimization Approach to Supply Chain Management. IPCO 2004: pp. 86-100.

[6] C. Bischof and A. Griewank, Computational differentiation and multidisciplinary design, in Inverse Problems and Optimal Design in Industry, H. Engl and J. McLaughlin, eds., Stuttgart, Teubner Verlag, 1994, pp. 187--211.

[7] C. Chandra, and J. Grabis, Supply Chain Configuration Using Simulation Based Optimization, Proc. of 13th Annual Industrial Engineering Research Conference, May 15-18, 2004, Houston, Texas.

[8] C. Chandra, S. Kumar, and A. V. Smirnov, E-Management of Scalable Cooperative Supply Chains: Conceptual Modeling and Information Technologies Framework, Human Systems Management, Vol. 20, No. 2, pp. 83-94, 2001.

[9] Chevalier A. La programation dynamique. Dunod Décision, 1977.

[10] W. Dingwei, F. Shu-Cherng, and H.L.W. Nuttle, Soft Computing for Multi-Customer Due-Date Bargaining, IEEE Transactions on Systems, Man and Cybernetics, Part (c), vol. 29(4), pp. 566-575, November 1999.

[11] J. Eschenbächer, P. Knirsch and I. J. Timm, Demand Chain Optimization by Using Agent Technology, in Proc. of the IFIP WG 5.7 International Conference on Integrated Production Management, Tromso, Norway, June 28-30, pp. 285-292, 2000.

[12] N. Gaither and G. Frazier, Operations Management, Southwestern Thomson Learning, Cincinnati, Ohio, 2002.

[13] W. Hoover, E. Eloranta, J. Holmström and K. Huttunen, Managing the Demand-Supply Chain: Value Innovations for Customer Satisfaction, John Wiley & Sons, New York, 2001.

[14] N. Julka, R. Srinivasan, I. Karimi, Agent-based supply chain management-1: framework, Computers & Chemical Engineering, vol. 26, no. 12, pp. 1755 – 1769, 2002.

[15] M. Littman and J. Boyan, A distributed reinforcement learning scheme for network routing, in J. Alspector, R. Goodman, and T. X. Brown, eds., Proc. of the 1993 International Workshop on Applications of Neural Networks to Telecommunications, pp. 45--51. Lawrence Erlbaum Associates, Hillsdale NJ, 1993.

[16] T. M. Mitchell, Machine Learning. . McGraw-Hill, 1997.

[17] F. Nah, M. Rosemann and E. Watson, Guest editorial: E-business Process Management, Business Process Management Journal, 10(1), pp. 1-15, 2004.

[18] W. B. Powell, J. A. Shapiro and H. P. Simão, An Adaptive Dynamic Programming Algorithm for the Heterogeneous Resource Allocation Problem, Transportation Science, vol. 36, no. 2, pp. 231 –249, May 2002.

[19] Production-scheduling Excel Toolbox, available at: http://www.production-scheduling.com/index.asp

[20] M.L. Puterman, Markov Decision Processes. Wiley-Interscience publication, 1994.

[21] L.E. Rocha-Mier, Learning in a Neural Collective Intelligence: Internet packet routing application, PhD Dissertation, National Polytechnic Institute of Grenoble (INPG), France, 2002.

[22] M. Sakarovitch, Optimisation Combinatoire. Hermann, 1984.

[23] L. Sheremetov, L. Rocha, I. Batyrshin. Towards a Multi-agent Dynamic Supply Chain Simulator for Analysis and Decision Support. IEEE NAFIPS-05 Annual Conference: Soft Computing for Real World Applications Ann Arbor, Michigan, June 22-25, 2005.

[24] A. Smirnov, L. Sheremetov, N. Chilov, and J. Romero-Cortes, Soft-computing Technologies for Configuration of Cooperative Supply Chain, Int. J. Applied Soft Computing. Elsevier Science, vol. 4(1), pp. 87-107, 2004.

[25] J.M. Swaminathan, S.F. Smith, and N.M. Sadeh, Modeling Supply Chain Dynamics: A Multiagent Approach, Decision Sciences, 29(3), pp. 607-631, 1998.

[26] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, The MIT Press, 1998.

[27] W. Wang, J. Ryu, D.E. Riv era, K.G. Kempf and K.D. Smith, A Model Predictive Control Approach for Managing Semiconductor Manufacturing Supply Chains under Uncertainty, Annual AIChE Meeting, San Francisco, CA, Nov. 16-21, 2003, pp. 1-34.

[28] C. Watkins, Learning from Delayed Rewards. Ph D Dissertation, Cambridge University, 1989.

[29] D. Wolpert and T. Kagan, An introduction to collective intelligence, Technical Report NASA-ARCIC-99-63, NASA Ames Research Center, 1999.

[30] Q. Zhang, M.A. Vonderembse, and J.S. Lim, Manufacturing flexibility: defining and analyzing relationships among competence, capability, and customer satisfaction, J. of Operations Management, vol. 21, pp. 173-191, 2003.

Figure captions

Fig. 1 Supply Chain Generic Scheme

Fig. 2 Generic view of an agent within the COIN framework.

Fig. 3 Fuzzy multi-agent framework for supply chain modelling and optimisation – MASCS

Fig. 4 Simple three-echelon supply chain

Fig. 5 Message interchange among a set of PAs (Sniffer Agent screenshot).

Fig. 6 Generation of the membership functions to define echelon's capacity

Fig. 7 Original structure of the SCN

Fig. 8 Average production time: Q-routing (Shortest Path) Original SC, Q-Neural Original SC, Q-Neural Final SC

Fig. 9 Average inventory levels: Q-Neural Original SC, Q-Neural Final SC

Fig. 10 Final adjusted structure of the SCN