Decentralized Optimization of Distributed Supply-Chain

Marco Ghirardi, Giusepppe Menga, Nicola Sacco

Abstract— In this paper, the problem of optimizing the performances of a distributed supply chain is faced by means of a decentralized architecture. In particular, a general scheme for an optimization algorithm, based on Lagrangian relaxation of the precedence constraints, is presented. In this framework, the trade-off between the production cost of each node of the supply-chain and the cost of the whole network, which consists of earliness and tardiness of the received orders, is taken into account. Finally, a study case is analyzed in detail and some simulation results are presented.

I. INTRODUCTION

In the last decades, due to the new technologies in the fields of information and telecommunication, global automation has transferred and extended classical process control and factory automation ideas to geographically distributed environments. In this framework, distributed supply chains result to be networks of autonomous components nodes, operating in a competitive or cooperative environment, in which no hierarchy in decision making is enforced and where the initiatives to reach a common goal is taken by each partner.

While in the relevant literature there are many centralized approaches to the optimization of integrated production - transportation systems by means of integer programming (i.e., [1]), or partially distributed algorithms based on Lagrangian relaxation ([2]), in the era of Internet many researchers devoted their attention to the interconnections of autonomous agents (i.e., [3] or [4]) and decentralized decision modules which are able to guarantee that no hierarchy in the decision making is enforced.

While it is unreal to think of that all the node of supply-chain are capable or, more often, want to share information about their production planning with the other nodes, especially in a competitive environment, even in a cooperative environment sharing and exchanging a lot of information may require a significant amount of time.

To cope with these problems, in this paper the problem of optimizing the performances of a distributed supply-chain is faced by means of a distributed algorithm which allow to reach good performances by minimizing the information exchanged among the nodes of the supply chain.

In this framework the main effort of this work is to formalizing an effective decentralized problem which take into account both the production costs of each element of the supply-chain and the cost due to earliness and tardiness of the received orders.

All the authors are with Department of Control and Computer Sciences, Polytechnic of Turin, 10129 Turin, Italy {marco.ghirardi, giuseppe.menga, nicola.sacco}@polito.it

II. MODELING ISSUES

In this section, the model of the considered distributed supply-chain is presented with the aim of describing the interaction between the nodes and of introducing the considered performance measures. In particular, after describing the considered supply-chain structure, the dynamic behavior of the external demand and of the supply-chain is presented.

A. The Supply-Chain Structure

Consider the supply-chain depicted in Fig. 1 where many different nodes, which represent suppliers, assemblers, etc., interact with each other.

Formally, in this model the elements of such a net belong to the sets $S = \{s_1, \ldots, s_8\}$ of $S$ suppliers, which provide raw materials to the net, $A = \{a_1, \ldots, a_A\}$ of $A$ assemblers, $M = \{m_1, \ldots, m_M\}$ of $M$ manufacturers, which perform some kind of activity on the parts, $T = \{t_1, \ldots, t_T\}$ of $T$ transportation systems, which manage the movement of the parts in the net, and $E = \{e_1, \ldots, e_E\}$ of $E$ end users.

Moreover, in this general formulation each node $i$ of the sets $A$, $M$, and $T$, may be both input or output of the other nodes while, on the other hand, the suppliers are only inputs nodes of the supply-chain (i.e., they don’t have any upstream node), and the end users are only output of the network.

Then, to the aim of simplifying the notation, the sets $IN_i$ and $OUT_i$, which gathers the indices of the input and output nodes of any generic element $i$, respectively, are introduced. In this notation, the indices gathered in the set $IN_i$ are associated with elements of the sets $S$, $A$, $M$, and $T$, while the indices gathered in the set $OUT_i$ are associated with elements of the sets $A$, $M$, $T$, and $E$.

In such a general formulation the nodes of the supply-chain can cooperate to reach a shared common goal or not. However, from point of view of the structure here introduced it is not necessary to specify whether, for instance in the
supply-chain of Fig. 1 the nodes \( t_1 \) and \( t_2 \) cooperate by sharing the demand coming from \( a_1 \) or compete by trying to satisfy the whole demand of \( a_1 \) on their own.

Finally, each node of the net is characterized by an inventory for the incoming parts and an inventory for outgoing parts, which usually represent costs, in general different for the two inventories, for the node.

Note that due to this inventories, in the most general formulation it is possible to easily take into account “fluidified” productions, by considering demands of production of batches instead of parts.

B. Dynamic behavior of the demand

In this section, the dynamic behavior of the demand that each node can receive or generate is described.

Then, consider the generic node \( i \) and let \( d_{i,h}(t), h \in \text{OUT}_i \) and \( d_{i,l}(t), l \in \text{IN}_i \), be the demand for the node \( i \) generated by its output node \( h \) and the demand generated by the node \( i \) for its input node \( l \), respectively. Such demands consists of requests of finished jobs characterized by a due date.

Furthermore, let \( t_0 \) be the actual time instant, the time instant before whom the node \( i \) can not to schedule new a production.

Thus, at \( t_0 \), the total forecasted production demand the node \( i \) has to satisfy is given by

\[
d^\text{tot}_i(t) = \sum_{h \in \text{OUT}_i} d_{i,h}(t), \quad t \geq t_0.
\]

Note that the demand \( d^\text{tot}_i(t) \) can dynamically change while time passes since certain orders \( \delta_i(t) \) and \( \eta_i(t) \) can be added and removed, respectively, to the actual forecasted demand, giving

\[
d^\text{tot}_i(t) = d^\text{tot}_i(t) + \delta_i(t) - \eta_i(t), \quad t \geq t_0,
\]

as represented in Fig. 2.

C. The performances indices

To the aim of introducing the performance indices of a distributed supply chain, consider again the single node \( i \) of a generic supply-chain. It is clear that, if such a node is not in some way “forced” in taking into account a goal shared with the other elements of the net, for instance by means of additional constraints or by means of additional terms in its cost function, it simply tries to minimize its own costs.

On the other hand, the whole supply-chain often has an overall goal which might consist, for instance, of satisfying all the external demands coming from the end users by minimizing the cost of earliness and tardiness, or finding a “flexible” production plan which can quickly schedule any new external order. Unfortunately, in most cases, such global and local objectives are in opposition since minimizing the global earliness and tardiness requires high inventory levels (which as said represent costs for the single nodes) or since adding new orders in an already scheduled plan can introduce additional terms in the cost function of each single element.

Furthermore, as mentioned in Sec. II-A, there are two ways the nodes can interact with each other:

1) they can share a common goal and cooperate to reach it. In this cooperative case some elements of the supply-chain might “pay” higher production costs rather than if they were alone while the whole systems has a global benefit, for instance in terms of earliness and tardiness with respects of the orders of the end users;
2) the whole system has system still has a goal but the elements of the net compete with each other trying to maximize their own benefit. A typical example of this competitive case, is when two or more elements of a supply-chain tries to satisfy on their own a demand coming from a common output node.

III. THE OPTIMIZATION PROBLEM

In this section the problem of finding a good production planning for a distributed supply is faced by means of a decentralized algorithm, based on the lagrangian relaxation of the precedence constraints. Moreover, the problem of guaranteeing the boundedness of the distributed optimization problem is faced.

A. The Contract Protocol

In this section the protocol of interaction between the nodes of a distributed supply chain, based on the classical Contract Net Protocol (see [5]) is described. To this aim, consider the basic scheme represented in Fig. 3 where, for the sake of simplicity and without loosing generality, a generic

---

**Fig. 2.** A) A generic demand of the \( i \)-th node of the net; B) Added and removed orders for the \( i \)-th node; C) The new total demand of the \( i \)-th node.
node $i$ is represented with only an input and an output node together with the messages exchanged by the nodes.

In this framework, when the node $i$ is asked by its output node $(i + 1)$, (i.e. $i + 1$ is in $\text{OUT}_i$), to process a job $j$, the CNP evolves as follow:

Algorithm 3.1: (Convergence algorithm)
1) $k = 0$;
2) given the Requested Due Dates (RDD) $R^j_{i,i+1}(k)$ of each job $j$ (which represents the due dates of the job $j$) from the customer $i + 1$ and the Tentative Due Date (TDD) $T^j_{i-1,i}(k)$ of each job $j$ (which represents the release time of the job $j$) from supplier $i - 1$, the node $i$ solves its local optimization problem, hence obtaining the completion times of each job on node $i$;
3) the node $i$ proposes a new Tentative Due Date $T^j_{i,i+1}(k)$ (which coincides with the completion time of $j$) to the customer $i + 1$;
4) node $i$ proposes a new Requested Due Date $R^j_{i-1,i}(k)$ (which coincides with the starting time of $j$) to the supplier $i - 1$;
5) at next iteration (setting $k = k + 1$), the node $(i + 1)$ proposed a new RDD and the node $(i - 1)$ proposed a new TDD and the process restart from step 2. ■

Note that such an algorithm, in this form, does not ensure that the contract process will converge since there are no ways to force the nodes in modifying the requested or proposed due dates that they consider to be optimal.

B. The Lagrangian relaxation

In order to apply Alg. 3.1 problems related to each single element of the supply chain must be defined. To this aim, the lagrangian relaxation technique is used since Lagrange multipliers can be used in order to move constraints from a constrained optimization problem into the cost function (see for example [6] or [7]). In this formulation, the precedence constraints between the operations on the different elements of the chain are moved to the objective function of the optimization problem. More precisely, in this formulation, for the node $i$, the lagrangian relaxation consists of associating with the job $j$ the two lagrangians

1) $\lambda^j_{i-1,i}$ which, at each iteration $k$ weights in the local cost function the difference $R^j_{i-1,i}(k) - T^j_{i-1,i}(k)$;
2) $\lambda^j_{i,i+1}$ which, at each iteration $k$ weights in the local cost function the difference $R^j_{i,i+1}(k) - T^j_{i,i+1}(k)$.

Hence, the problem is decomposed into $m$ independent problems which are coupled two-by-two by the lagrangian multipliers values, the proposed due dates and tentative delivery dates.

In this framework, the lagrangian multipliers for each job are updated by a subgradient based method where the direction of the update is given by the normalized difference between the requested due date and the tentative due date

\[
\lambda^j_{i-1,i}(k + 1) = \lambda^j_{i-1,i}(k) + \alpha^j_{i-1,i}(k) \frac{R^j_{i-1,i}(k) - T^j_{i-1,i}(k)}{|R^j_{i-1,i}(k) - T^j_{i-1,i}(k)|}
\]

\[
\lambda^j_{i,i+1}(k + 1) = \lambda^j_{i,i+1}(k) + \alpha^j_{i,i+1}(k) \frac{R^j_{i,i+1}(k) - T^j_{i,i+1}(k)}{|R^j_{i,i+1}(k) - T^j_{i,i+1}(k)|}
\]

being $\alpha^j_{i,j}(k)$ step coefficients which depend on the job $j$, on the couple of nodes $(i - 1, i)$ or $(i, i + 1)$, and usually decreasing with $k$.

Such a relaxation acts only on the precedence constraints and does not depend on the kind of problem of each single node of the supply chain which can be equivalently, for instance, a single machine or a parallel machines scheduling problem.

Note also that the results obtained by the multiplier update technique depends on the order in which the different elements of the system update the multipliers.

Furthermore, due to the relaxation of the precedence constraints, the solution obtained by the subgradient algorithm may be infeasible for the overall supply chain problem, i.e., it is not ensured that $R^j_{i,i+1}(k) \geq T^j_{i,i+1}(k)$ for every job $j$. In this situation not all the jobs can start their processing at the scheduled time. Following a rolling horizon strategy, at each time $t$, all the enabled jobs, i.e., the jobs for which the actual release time $r_{i,j}$ is less than $t$, can start their processing. In this framework, while time passes, each new iteration of the optimization algorithm consider the new updated real state of the system, thus taking into account that some jobs can not be rescheduled.

IV. STUDY CASE

In this section, the convergence of the lagrangian relaxation based method is discussed by means of a simplified study case. In addition, it will be shown that there exists a region in the plane $(\lambda^j_{i,i+1}, \lambda^j_{i-1,i})$ in which the multipliers must stay, for each step $k$ of the algorithm, to the aim of guaranteeing the boundedness of each single node optimization problem.

Note that since the lagrangians only depend on the precedence constraints, the boundedness of the distributed optimization algorithm, does not depend on kind of production of each single node.

Let us introduce the following notation:

- $W^L_{i,j}$ waiting time for job $j$ in the inventory of the incoming parts of the node $i$;
- $\omega^L_{i,j}$ weight of $W^L_{i,j}$ in the cost function;
- $W^R_{i,j}$ waiting time for job $j$ in the inventory of the outgoing parts of the node $i$;
- $W_{i,j} = W^L_{i,j} + W^R_{i,j}$ waiting time of job $j$ between operations on machine $i$ and $i + 1$;
\[ \min \sum_{j} \left[ \sum_{i=1}^{m-1} (\omega_{i,j} W_{i,j}) + \omega_{j} E_{m,j} + \omega_{i} T_{m,j} \right] \] (1)

subject to

\[ c_{1,j} \geq \text{dur}_{1,j}, \forall j \in J \] (2)

\[ c_{i-1,j} - c_{i,j} + W_{i,j} + \text{dur}_{i,j} = 0, \forall j \in J, \forall i \in 2, \ldots, m \] (3)

\[ c_{i,h} - \text{dur}_{i,h} \geq c_{i,j} \wedge \text{dur}_{i,j} \geq c_{i,h}, \forall j, h \in J, \forall i \in 1, \ldots, m \] (4)

\[ c_{m,j} + E_{m,j} - T_{m,j} = \text{dd}_{m,j}, \forall j \in J \] (5)

where the constraints (2) prevent processing of jobs before their ready times (supposed equal to 0) on machine one, the constraints (3) are the precedence constraints for the operations on job \( j \) between machine \( i-1 \) and \( i \) and define the waiting times between these machines, the constraints (4) are the machine capacity constraints which state that each machine can process only one job at a time, and, finally, the constraints (5) define earliness and tardiness for machine \( m \). The objective function (1) is the sum of all the waiting times, tardiness and earliness costs of the schedule. Note that from the point of view of the particular problem here addressed, while flow shop problems with earliness/tardiness costs have been extensively studied in literature, there are few works on the problem which includes the intermediate inventories costs. In effect, most of them face the simplified problem with common due dates, while a centralized approach for the problem faced in this work can be found, among others, in [8].

In order to decompose this model into \( m \) different optimization problems, consider these observations:

- in an optimized and collaborative environment between machine \( i-1 \) and \( i \) only one between inventories \( W^{R}_{i,j} \) and \( W^{L}_{i+1,j} \) will be used (the less expensive). Hence, we can suppose that machine \( i-1 \) 'see' the same holding cost \( (\omega_{i,j} = \omega_{i+1,j}) \) and share the cost of the inventory in equal parts;
- for each machine \( i = 1, \ldots, m-1 \), the starting time \( c_{i+1,j} - \text{dur}_{i+1,j} \) of job \( j \) on machine \( i+1 \) represents the requested due date \( R_{i+1,j}^{L} \) of job \( j \), while for each machine \( i = 2, \ldots, m \), the completion time \( c_{i-1,j} \) of job \( j \) on machine \( i-1 \) represents the tentative delivery time \( T_{i-1,j}^{L} \) proposed by machine \( i-1 \);
- the only constraints which couple the problems on the different machines are the precedence constraint (3). These constraints can be rewritten as:

\[ c_{i,j} \geq T_{i-1,j}^{L} + \text{dur}_{i,j} \]

\[ W_{i,j}^{L} \geq c_{i,j} - \text{dur}_{i,j} - T_{i-1,j}^{L} \]

\[ W_{i,j}^{R} \geq R_{i+1,j}^{L} - c_{i,j} \]

The first of these equations is the precedence constraint while the second and the third are redundant constraints for the original problem, but ensure that a measure of the waiting time between machines will be computed in the relaxed problems.

For the sake of simplicity, let us define \( \lambda_{i,j}^{L}(k), \forall j \), as the lagrangian multiplier associated to the relaxation of the precedence constraint of job \( j \) between machine \( i-1 \) and \( i \), \( i = 2, \ldots, m \), and \( \lambda_{i,j}^{R}(k), \forall j \), as the lagrangian multiplier associated to the relaxation of the precedence constraint of job \( j \) between machine \( i-1 \) and \( i \), \( i = 1, \ldots, m-1 \). Note that within each consecutive machine \( i-1 \) and \( i+1 \), \( \lambda_{i,j}(k) = \lambda_{i+1,j}(k), \forall j \). Such multipliers is used to relax the precedence constraints: in the relaxed problem a job can start an operation before the previous one is completed.

The result of the relaxation is described in the next paragraphs where the step counter \( k \) is dropped with the aim of simplifying the notation.

Consider first node 1 of the supply chain (i.e., the supplier) whose cost function only consists of the inventory cost of its outgoing parts inventory and of the cost associated to the relaxed constraint:

\[ \min \sum_{j \in J} \omega_{i,j}^{R} W_{i,j}^{R} + \lambda_{1,2}^{L} (c_{1,j} - R_{1,2}^{L}) \] (6)

subject to

\[ W_{1,j}^{R} \geq R_{1,2}^{L} - c_{1,j}, \forall j \in J \]

\[ c_{1,h} - \text{dur}_{1,h} \geq c_{1,j} \wedge \text{dur}_{1,j} \geq c_{1,h}, \forall j, h \in J \]

\[ c_{1,j} \geq \text{dur}_{1,j}, \forall j \in J \]

Furthermore, consider the generic internal node \( i, i = 2, \ldots, m-1 \), of the net depicted in Fig. IV-A and its cost
function
\[
\min \sum_{j \in J} \left[ \omega_{ij}^L W_{ij,1}^L + \omega_{ij}^R W_{ij,1}^R + \lambda_{i+1,j}^L (c_{i,j} - R_{i+1}^j) + \lambda_{i-1,j}^L (T_{i-1}^j - c_{i,j} - d_{i,j}) \right]
\]
and its constraints
\[
\begin{align*}
W_{ij,1}^L & \geq c_{i,j} - d_{i,j} - T_{i-1}^j, \quad \forall j \in J \\
W_{ij,1}^R & \geq R_{i+1}^j - c_{i,j}, \quad \forall j \in J \\
c_{i,h} - d_{i,j} & \geq c_{i,j} \land c_{i,j} - d_{i,j} \geq c_{i,h}, \quad \forall j, h \in J \\
c_{i,j} & \geq d_{i,j}, \quad \forall j \in J
\end{align*}
\]
which describe the problem of minimizing the inventory levels for ingoing and outgoing parts and the cost associated to the relaxed relaxed constraint.

Finally, the last node $m$ of the supply chain in Fig. IV-A has a cost function consisting of the level of its incoming parts inventory and of earliness and tardiness with respect of the job due dates
\[
\min \sum_{j \in J} \left[ \omega_{mj}^L W_{mj,1}^L + \omega_{mj}^R W_{mj,1}^R + \lambda_{m,n}^L (T_{m-1,n} - c_{m,j} - d_{m,j}) \right]
\]
subject to
\[
\begin{align*}
E_{m,j} & = \max\{0, dd_{m,j} - c_{m,j}\}, \quad \forall j \in J \\
T_{m,j} & = \max\{c_{m,j} - dd_{m,j}, 0\}, \quad \forall j \in J \\
W_{m,j}^L & \geq c_{m,j} - d_{m,j} - T_{m-1,1}, \quad \forall j \in J \\
c_{m,h} - d_{m,j} & \geq c_{m,j} \land c_{m,j} - d_{m,j} \geq c_{m,h}, \quad \forall j, h \in J \\
c_{m,j} & \geq d_{m,j}, \quad \forall j \in J
\end{align*}
\]
Now the problem is decomposed into $m$ independent problems which are coupled two-by-two by the lagrangian multipliers values and the resulting proposed due dates and tentative delivery dates.

B. Boundedness region

In this section, a feasible area for the lagrangian multipliers are given, with the aim of keeping the local problems boundedness.

To this end, consider the local problem (6). In order for the optimal solution of this problem to be finite, all the coefficient of any completion time variable in the objective function must be positive. In our case, it is trivial to understand that
\[
\lambda_{i,2}^L < \omega_{i}^L, \quad \forall j \in J.
\]
Furthermore, by considering problem (7) we find:
\[
\begin{align*}
\lambda_{i-1,j}^L & < \lambda_{i,j}^L + \omega_{j,i+1}^R \\
\lambda_{i+1,j}^L & < \lambda_{i,j}^L \land \lambda_{i,j}^L + \omega_{j,i+1}^R \quad \forall j \in J
\end{align*}
\]
which keeps the problem bounded for $i = 2, \ldots, m - 1$. The region described by the equations (8) is depicted in light gray Fig. 5.

Finally, about the problem related to node $n$, it is easy to deduce
\[
\lambda_{n}^L < \omega_{n}^L, \quad \forall j \in J.
\]

C. Boundedness rule

In this section a distributed rule which keep the local problems bounded is stated. While the need of such rule comes from the fact that there is not any supervisor and each node have to apply a control by itself without affecting the behavior of the other nodes, in a distributed problem such this one, not all the nodes can be allowed to update the multipliers values at a time with the aim of guaranteeing the consistence of the multipliers. In effects, the lagrangian multipliers appearing in the local problem of the node $i$ can not change while the node $i$ is solving the problem.

Coming back to the lagrangian admissible vales, consider a generic node $i$ of the net and consider the plane of the points $(\lambda_{i-1,i}^L, \lambda_{i+1,i}^L)$ depicted in Fig. 5 where each the coordinates of each point represents the lagrangians associated with the job $j$ with respect to the node $i - 1$ and $i + 1$ respectively. Moreover let $d_{i-1}^L$ and $d_{i+1}^R$ be the distances between the two lines and the bisector and let $d_{i}^B$ be the distance between the point $(\lambda_{i-1,i}^L, \lambda_{i+1,i}^L)$ and the bisector.

Before solving its own local problem he has to “check” the bounds on the lagrangians and, eventually, update them by means of the following algorithms.

Then, to keep the problem bounded, before beginning its own optimization, the generic node $i$ perform the following algorithm:

\textbf{Algorithm 4.1: Stability Algorithm}

for $j = 1 : n$
1) if $\lambda_{i-1,i}^L \leq \lambda_{i,j}^L$ go to step 4;
2) if $d_{i}^B < d_{i-1}^L$ continue (there is no need to update the multipliers);
3) the updated lagrangians can be computed by means of
the following equations, depending on the node $i$. In particular:

a) if $i = 1$
\[ \hat{\lambda}_{1,2}^i = \omega_{1,2}^i - \varepsilon \] (9)

b) if $i = 2, \ldots, m - 1$
\[
\begin{align*}
\hat{\lambda}_{i-1,i}^j &= \frac{\lambda_{i,i+1}^j + \lambda_{i-1,i}^j + \omega_{i,j}^i}{2} - \frac{\varepsilon}{\sqrt{2}} \\
\hat{\lambda}_{i,i+1}^j &= \frac{\lambda_{i,i+1}^j + \lambda_{i+1,i}^j - \omega_{i,j}^i}{2} + \frac{\varepsilon}{\sqrt{2}}
\end{align*}
\] (10)
c) if $i = m$
\[ \hat{\lambda}_{m-1,m}^i = \omega_{m-1,m}^i - \varepsilon \]

which give a point $(\hat{\lambda}_{i-1,i}^j, \hat{\lambda}_{i+1,i}^j)$ which is in the stability area at a distance $\varepsilon$ from the right bound. Then continue;

4) if $d_{i}^j < d_{i+1}^j$ continue;

5) the updated lagrangians can be computed by means of the following equations, depending on the node $i$. In particular:

a) if $i = 1$ use equation (9)

b) if $i = 2, \ldots, m - 1$
\[
\begin{align*}
\hat{\lambda}_{i-1,i}^j &= \frac{\lambda_{i,i+1}^j + \lambda_{i-1,i}^j + \omega_{i,j}^i}{2} + \frac{\varepsilon}{\sqrt{2}} \\
\hat{\lambda}_{i,i+1}^j &= \frac{\lambda_{i,i+1}^j + \lambda_{i+1,i}^j - \omega_{i,j}^i}{2} - \frac{\varepsilon}{\sqrt{2}}
\end{align*}
\] (11)
c) if $i = m$ use equation (10)

which give a point $(\hat{\lambda}_{i-1,i}^j, \hat{\lambda}_{i+1,i}^j)$ which is in the stability area guaranteeing a “stability margin” $\varepsilon$.

D. Experimental results

In this section, some preliminary results relevant to the case study are presented with the aim of showing the main characteristics of the proposed methodology. A simulation of the study case has been implemented using C++ and X-Press solver libraries in order to optimally solve the local problems and a test instance with 6 jobs has been created. All the orders are known at the beginning of the time horizon, fixed to 500 iterations of the subgradient algorithm. At each iteration we evaluate the objective function of the current overall solution in order to create a graph of the evolution of the optimization, shown in Fig. 6. It is possible to see that in less than 50 iterations the algorithm is able to improve the objective function of the solution, and then begins to bounce trying to find out new solutions. We also computed the optimal solution for the considered instance of the problem, which is 125 (shown with a horizontal line on the graph): the distributed algorithm is able to reach the optimal solution after about 430 iterations. About the CPU time needed to solve the problem, the centralized optimal solution (obtained with X-Press solver) has been obtained in 12 minutes, while the simulation of the decentralized solution needed only 75 seconds. This is what we expected: the decentralized algorithm solves only one-machine scheduling problems, which are much easier than a 5-machine flow shop.

V. Conclusion

In this paper a model and an optimization algorithm for distributed supply chain has been presented. In addition, a discussion about the stability of the algorithm has been presented and a suitable control law has been proposed. At the moment work in progress are about the realization of more realistic case studies, including the definition of models for transportation nodes and assembler nodes and rules for handling competitive suppliers. Another issue is about the definition of rules for the synchronization of the distributed algorithm, in order to improve its performances in terms of solution quality for instances with a realistic number of orders.

REFERENCES