Hindawi Publishing Corporation
Modelling and Simulation in Engineering
Volume 2011, Article ID 603632, 10 pages
doi:10.1155/2011/603632

Research Article

Stability of the Supply Chain Using System Dynamics Simulation and the Accumulated Deviations from Equilibrium

Luis Rabelo,[1] Alfonso T. Sarmiento,[1] and Albert Jones[2]

[1] *Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816, USA*
[2] *Manufacturing Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA*

Correspondence should be addressed to Luis Rabelo, luis.c.rabelo@nasa.gov

Received 24 August 2010; Revised 23 January 2011; Accepted 18 February 2011

Academic Editor: Hing Kai Chan

We propose and demonstrate a new methodology to stabilize systems with complex dynamics like the supply chain. This method is based on the accumulated deviations from equilibrium (ADE). It is most beneficial for controlling system dynamic models characterized by multiple types of delays, many interacting variables, and feedback processes. We employ the classical version of particle swarm optimization as the optimization approach due to its performance in multidimensional space, stochastic properties, and global reach. We demonstrate the effectiveness of our method based on ADE using a manufacturing-supply-chain case study.

1. Introduction

The current world economic climate has led to highly volatile markets and fluctuating demands for manufactured goods. Consequently, demand forecasting and inventory control have become major concerns for supply chain managers. In recent papers [1–3], the authors demonstrated that the classical supply-chain approach to forecasting future inventory demand [4] is not enough to avoid huge swings in finished goods and in-process inventories. The authors implied that a new approach to modeling supply chain behavior based on a deep understanding of the structure of the problem, the causalities among the system variables, and the different types of time delays is needed. This echoes results from [5] who said "behavioral functions that underlie market actions can be more successfully interpreted through nonlinear decision functions" and "that theory should be drawn from actual causality relationships". Modeling such nonlinear decisions and causal relationships is the focus of systems dynamics (SD). We believe that system dynamics combined with elements from nonlinear control theory, calculus, and optimization can be used to understand, monitor, and stabilize supply-chain behavior.

System dynamics models of supply chains provide a mathematical interpretation of the intricate relationships between numerous variables. Sets of those variables can form feedback loops based on the causality structure of the problem. These intermingled feedback loops can act simultaneously, but at different times, they may have diverse effects. Therefore, transitions from a regime to another are widespread. Most system dynamics models used for supply chains are complex and nonlinear [6]. Nonlinearities due to the multiplying effect of multiple causal variables upon another variable are "quite common in SD and difficult to tackle" [7]. The number of variables and feedback loops, the discontinuities, and time delays increases the level of complexity [8]. Time delays being added to "negative feedback loops increases the tendency for the system to oscillate" as stated by Sterman [9]. Therefore, this level of complexity makes more difficult the task to design stable supply chains.

Instability is a major cause for poor performance in supply chains [4, 10]. Traditional methods in structural analysis can be used to investigate and stabilize supply chains using SD modeling. One of these methods is loop knockout [10]. Loop knockout is a brute force method that relies on testing various feedback loops. This analysis of the different feedback loops is used to visualize the long-term effects of changes that could be implemented in order to stabilize the supply chain. This method can be effective for very simple models. However, it is not enough for medium to higher levels of complexity. Another method is eigenvalue analysis. Eingenvalue analysis is used to characterize the modes of behavior [11–13]. The analysis is based on the linearization of the mathematical description at infinitesimal points and then study the evolution of the eigenvalues and their elasticities [14]. The utilization of this method cannot be easily generalized to nonlinear models. Furthermore, methods in structural analysis rely on sensitivity analysis to determine the parameter values of the stabilization policies. Therefore, we propose a new alternative to stabilize the supply chain modeled as a dynamic system based on the ADE.

According to [15], the concept of stability is linked to the notion of equilibrium points—"an equilibrium point (EP) is stable if all solutions starting at nearby points stay nearby; otherwise it is unstable". He also described the notion of asymptotic stability which occurs when all solutions tend to the EP as time goes to infinity. In nonlinear control theory, stability of equilibrium points can be determined by (1) linearization around the EP and the values of the associated eigenvalues or (2) Lyapunov functions [15] when linearization is inadequate to approximate the global behavior of the system [16].

Note, however, that it is not always easy to find or construct a Lyapunov candidate function for a specific system. Because of this, we propose the use accumulated deviations from equilibrium. It is very well known from optimization-based control theory that minimizing the deviations of controlled variables from some desired level can have the effect of generating stable solutions (see the Appendix). We have developed a methodology based on this idea—stability analysis based on the accumulated deviations from the equilibrium (SADE)—in order to optimize and stabilize nonlinear systems. The optimization engine for this methodology is particle swarm optimization (PSO).

We will use the SADE methodology together with the concept of asymptotic stability to minimize oscillatory behaviors of specific control (state) variables—such as in-process or finished goods inventory. If necessary, stability can be extended to the whole system by using a weighted average function that includes all state variables. This also allows higher weights to be assigned to those variables considered more important. Since our methodology does not require linearization of the system or eigenvalue calculations, it can be applied as a general procedure to linear or nonlinear dynamic systems. This method is novel from the perspective of SD modeling.

The paper is organized as follows. In Section 2, we describe the SADE methodology including the optimization problem and PSO implementation. In Section 3, we give details about the supply-chain case study and relevant systems dynamics literature. Those details include the definition of the optimization problem, procedures for testing policy robustness, and the respective comparison with a local search algorithm. We conclude this paper with further discussion of some key points and proposed future work.

2. Sade Methodology

2.1. Description. The SADE methodology and its general functioning are shown in Figure 1. The supply chain environment represents the actual participants, structure, strategies, policies, objectives, variables, constraints, and parameters of the real system. This environment captures all of different scenarios of the supply chain (SC) over time. These scenarios, together with the associated decisions, produce changes in the behavior of the supply chain. By behavior, we mean the observed patterns in the state variables over time.

The SD model replicates the dynamic behavior of the supply chain environment. We chose the SD modeling approach based on the advice in [5]—it can capture the causal relationships, feedback processes, and multiple time delays necessary to track accurately behavioral evolution of the system. As exogenous events occur in the SC environment, their impacts on the behavior of the system are predicted using the SD simulation. If those predicted impacts do not show any instability patterns, no actions are taken. Otherwise, a new management policy must be found to remove the instability or minimize its impact.

This new policy is found using a PSO algorithm. That algorithm modifies the set of parameters that constitute the current policy until the ADE is minimized. In every iteration of the algorithm, the parameter set is sent to the SD model in order to calculate, through simulation, the value of the ADE (objective function). Simulation is used due to the difficulty of solving the complex dynamic equations by analytical methods. The optimization problem and the PSO algorithm are described in Sections 2.2 and 2.3.

Once the best setting of parameters (stabilization policy) is obtained, then it is implemented in the actual supply chain.

2.2. Optimization Problem. The SD model can be described by an equation of the form $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p})$, where $\mathbf{x}(t)$ is the vector of state variables (dimension n) and $\mathbf{p}$ is a vector of adjustable parameters (dimension q) with lower and upper bounds $\mathbf{p}^L$ and $\mathbf{p}^U$, respectively.

We can formulate an optimization problem that will find the parameter vector $\mathbf{p}^*$ that causes the state variable x_s to become asymptotically stable (see the Appendix) around the equilibrium point $x_s^{\mathrm{eq}}(\mathbf{p}^*)$. We will find this optimal parameter vector by minimizing the ADE (see the Appendix for the mathematical definition of ADE) for predetermined time horizon T and making use of Theorem 1 (see the Appendix). That is, we will find the vector that makes ADE converge. The optimization problem is then stated as

$$\begin{aligned}
\underset{\mathbf{p}}{\text{minimize}} \quad & J(\mathbf{p}) = \sum_{s=1}^{m} \left\{ w_s \int_0^T \left| x_s(t) - x_s^{\mathrm{eq}} \right| dt \right\}, \\
& \qquad \text{where } w_s \geq 0, \sum_{s=1}^{m} w_s = 1, \\
\text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}) \\
& \mathbf{x}(0) = \mathbf{x}_0 \\
& \mathbf{p}^L \leq \mathbf{p} \leq \mathbf{p}^U \\
& \mathbf{x}(t) \in R^n, \mathbf{p} \in R^q, \mathbf{p}^L \in R^q, \mathbf{p}^U \in R^q, \\
& \qquad t \in [0, T].
\end{aligned} \tag{1}$$

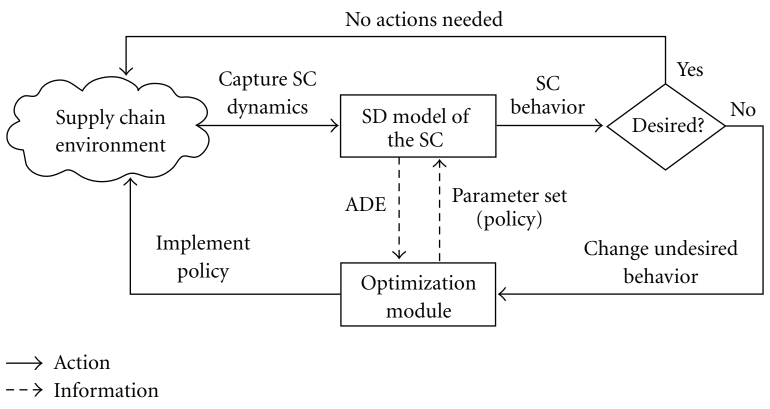

FIGURE 1: General procedure of the SADE methodology.

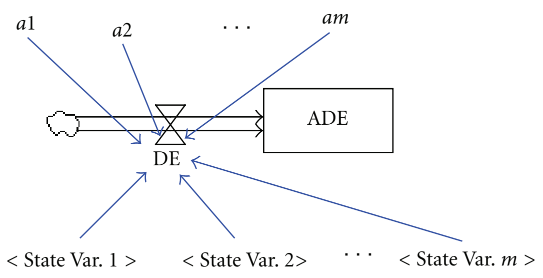

FIGURE 2: Stock and flow diagram for the objective function.

The objective function $J(\mathbf{p})$ is defined as the weighted average value of the ADE, and T is the time horizon. The use of weights, w_s, means that $J(\mathbf{p})$ will support the simultaneous stabilization of any subset of m state variables ($m \leq n$). This allows higher weights to be assigned to the variables that are considered more important.

If we do not know the equilibrium point x_s^{eq} in advance, we can modify $J(\mathbf{p})$ to include it as a variable (a_s) in the parameter vector. This step is supported by the results of Theorem 2 (see the Appendix).

The objective function defined in (1) can be incorporated very easily into any SD formulation by adding a "stock and flow" piece to the model that is linked to the state variables of interest as illustrated in Figure 2. Then, we define the variables DE and ADE as

$$\begin{aligned} \text{DE} &= w1^{*}\text{ABS (State Var. } 1 - a1) \\ &\quad + w2^{*}\text{ABS (State Var. } 2 - a2) + \cdots \\ &\quad + wm^{*}\text{ABS (State Var. } m - am), \\ \text{ADE} &= \text{INTEG (DE}, 0). \end{aligned} \tag{2}$$

2.3. PSO Algorithm. Particle swarm optimization (PSO) was invented in the mid-1990s by Kennedy and Eberhart [17]. PSO is conceptually simple and can be implemented in a few lines of code. In comparison with other stochastic optimization techniques like genetic algorithms (GAs) or simulated annealing, PSO has fewer complicated operations and fewer defining parameters [18]. PSO has also been shown to be more computationally efficient than GAs when applied to unconstrained nonlinear problems with continuous variables Hassan et al. [19].

The specific algorithm we use is called "local best PSO" [20]. This algorithm is based on a social network composed of neighborhoods related to each particle. The algorithm maintains a swarm of particles, where each particle represents a candidate solution to the optimization problem. These particles move across the search space communicating good positions to each other within the neighborhood and adjusting their own position and velocity based on these good positions. For this purpose, each particle keeps a memory of its own best position found so far and the neighborhood best position among all the neighbor particles. The goodness of a position is determined by using a fitness function. A typical stopping condition of the algorithm is when the maximum number of iterations has been exceeded. The basic elements of the algorithm are defined as follows.

(i) *Particle.* A particle i is represented by a n_p-dimensional real-valued vector $\mathbf{p}_i$. This vector is composed of particle positions p_{ij}; that is, $\mathbf{p}_i = [p_{i1}, p_{i2}, \ldots, p_{in_p}]$. Each particle position corresponds to one of the parameters of the parameter vector defined in the optimization Section 2.2.

(ii) *Swarm Size.* It is the number of particles in the swarm, and it is denoted by N.

(iii) *Fitness Function.* It is a mathematical function used to quantify how good the solution represented by a particle is. For a particle i, the fitness function is the objective function $J(\mathbf{p}_i)$ as defined in Section 2.2.

(iv) *Personal Best Position.* As a particle moves through the search space, it compares its fitness value at the current position to the fitness value it has ever attained so far, which is called the personal best position. For each particle i, the personal best position can be expressed as the real-valued vector $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{in_p}]$, and it is determined so that $J(\mathbf{y}_i) \leq J(\mathbf{p}_i), i = 1, \ldots, N$.

(v) *Neighborhood Size.* Defines the extent of the social iteration within the swarm [20]. Selection of neighborhoods was done based on particle indexes. Each particle has a neighborhood associated to, where B_i defines the set of indexes for the neighbors of particle i.

(vi) *Neighborhood Best Position.* It is the best position among all the personal best positions in the neighborhood. It is denoted by the real-valued vector $\hat{\mathbf{y}}_\mathbf{i} = [\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{\text{in}_p}]$, and it is determined so that $J(\hat{\mathbf{y}}_i) \le J(\mathbf{y}_j)$, $j \in B_i$.

(vii) *Global Best Position.* It is the best position among all the personal best positions achieved so far in the entire swarm. It is denoted by the real-valued vector $\mathbf{g} = [g_1, g_2, \ldots, g_{n_p}]$, and it is determined so that $J(\mathbf{g}) \le J(\mathbf{y}_i)$, $i = 1, \ldots, N$.

(viii) *Particle Velocity.* It is the velocity of the moving particle i represented by the real-valued vector $\mathbf{v}_\text{i} = [v_{i1}, v_{i2}, \ldots, v_{\text{in}_p}]$. This vector reflects both the experiential knowledge of the particle and socially exchanged information from the particle's neighborhood [20]. The experiential knowledge of a particle is generally referred as the *cognitive component*, which quantifies the performance of particle i relative to past performances. It is represented by the term $c_1\mathbf{r}_1(\mathbf{y}_i - \mathbf{p}_i)$. The socially exchanged information is referred as the *social component* of the velocity equation. It is represented by the term $c_2\mathbf{r}_2(\hat{\mathbf{y}}_i - \mathbf{p}_i)$.

(ix) *Acceleration Coefficients.* The acceleration coefficients, c_1 and c_2, together with the random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, control the stochastic influence of the cognitive and social components on the overall velocity of a particle [20]. The constants c_1 and c_2 are also referred to as trust parameters, where c_1 expresses how much confidence a particle has in itself, while c_2 expresses how much a particle has in its neighbors. The random vectors are defined as $\mathbf{r}_1 = [r_{11}, r_{12}, \ldots, r_{1n_p}]$ and $\mathbf{r}_2 = [r_{21}, r_{22}, \ldots, r_{2n_p}]$, where r_{1j} and r_{2j} are uniformly distributed random numbers in $[0, 1]$.

(x) *Inertia Weight.* It is a parameter "w" that is used to control the influence in the new velocity of a particle by its previous velocity (flight direction). Thus, it influences the tradeoff between the global and local exploration abilities of the particles [21]. For initial stages of the search process, where global exploration is required, it is recommended to set a large inertia weight, while for the last stages, the inertia weight should be reduced for better local exploration. A decrement function for decreasing the inertia weight at the iteration k can be given by $w(k) = \alpha w(k')$, where $\alpha = 0.98$ and k' is the last iteration when the inertia changed. A parameter "iteration lag" is defined to set the number of iterations since the last change in the fitness function that are required to change the inertia weight.

The steps of the algorithm are described in the following lines.

Step 1. Initialization

(i) Set iteration $k = 0$.

(ii) Generate N particles $\mathbf{p}_i(0) = [p_{i1}(0), p_{i2}(0), \ldots, p_{\text{in}_p}(0)]$, $i = 1, \ldots, N$, where $p_{ij}(0)$ is randomly selected according to a uniform distribution in the interval $[p_j^L, p_j^U]$, $j = 1, \ldots, n_p$.

(iii) Generate velocities $\mathbf{v}_i(0) = [v_{i1}(0), v_{i2}(0), \ldots, v_{\text{in}_p}(0)]$, $i = 1, \ldots, N$, where $v_{ij}(k)$ is randomly selected according to a uniform distribution in the interval $[0, v_{\max,j}]$, $j = 1, \ldots, n_p$, where $v_{\max,j} = p_j^U - p_j^L$.

(iv) Evaluate the fitness of each particle using $J(\mathbf{p}_i(0))$, $i = 1, \ldots, N$.

(v) Set the initial value of the personal best position vector as $\mathbf{y}_i(0) = \mathbf{p}_i(0)$, $i = 1, \ldots, N$.

(vi) Determine the neighborhood best position vector $\hat{\mathbf{y}}_i(0)$ using the formula $J(\hat{\mathbf{y}}(0)) = \min\{J(\mathbf{y}_j(0))\}$, $j \in B_i$.

(vii) Determine the global best position $\mathbf{g}(0)$ using the formula $J(\mathbf{g}(0)) = \min\{J(\mathbf{y}_i(0))\}$, $i = 1, \ldots, N$.

(viii) Set the initial value of the inertia weight $w(0)$.

Step 2. Iteration updating: set $k = k + 1$.

Step 3. Weight updating: if the fitness function has not decreased in a number of iterations equal to the "iteration lag," then update the inertia weight using $w(k) = \alpha w(k')$.

Step 4. Velocity updating: calculate the velocity of particle i by using

$$\begin{aligned}\mathbf{v}_i(k) = {} & w(k)\mathbf{v}_i(k-1) + c_1\mathbf{r}_1(k)[\mathbf{y}_i(k) - \mathbf{p}_i(k)] \\ & + c_2\mathbf{r}_2(k)[\hat{\mathbf{y}}_i(k) - \mathbf{p}_i(k)].\end{aligned} \tag{3}$$

Step 5. Position updating: based on the updated velocities, each particle changes its position according to the following equation:

$$\mathbf{p}_i(k) = \mathbf{v}_i(k) + \mathbf{p}_i(k-1). \tag{4}$$

Step 6. Personal best updating: determine the personal best position visited so far by each particle.

(i) Evaluate the fitness of each particle using $J(\mathbf{p}_i(k))$, $i = 1, \ldots, N$.

(ii) Set

$$\mathbf{y}_i(k) = \left\{ \begin{cases} \mathbf{y}_i(k-1) & \text{if } J(\mathbf{p}_i(k)) \ge J(\mathbf{y}_i(k-1)), \\ \mathbf{p}_i(k) & \text{if } J(\mathbf{p}_i(k)) < J(\mathbf{y}_i(k-1)). \end{cases} \right. \tag{5}$$

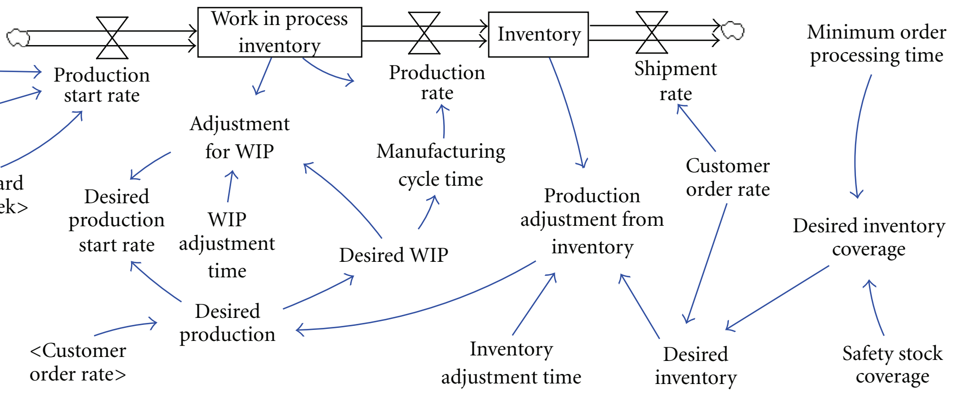

Figure 3: Inventory sector.

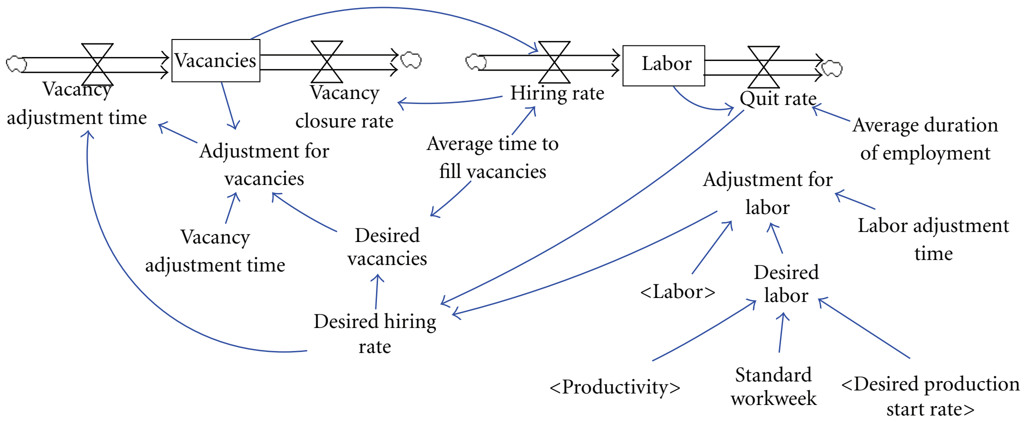

Figure 4: Labor sector.

Step 7. Neighborhood best updating: determine the neighborhood best position $\hat{\mathbf{y}}_i(k)$ visited so far by the whole swarm by using the formula

$$J(\hat{\mathbf{y}}_i) = \min\left\{J\left(\mathbf{y}_j\right)\right\}, \quad j \in B_i. \tag{6}$$

Step 8. Global best updating: determine the global best position $\mathbf{g}(k)$ visited so far by the whole swarm by using the formula

$$J(\mathbf{g}(k)) = \min\{J(\mathbf{y}_i(k))\}, \quad i = 1,..,N. \tag{7}$$

Step 9. Stopping criteria: if the maximum number of iterations is achieved, then stop, $\mathbf{g}^* = \mathbf{g}(k)$ is the optimal solution; otherwise, go to Step 2.

2.4. Differences between Our Approach and Current PSO Applications in Supply Chain Management. The contribution of our approach is the development of the ADE as objective function in order to solve the problem of instabilities in the supply chain. PSO has been selected as the mechanism to find a solution. The current PSO applications in supply chain management are very different from our approach. The current PSO applications emphasized straight forward solutions to static supply chain problems without regards to stability, robustness, and system dynamics [22–26].

2.5. Comparisons with Other Optimization Algorithms. Initial comparisons were performed against Genetic Algorithms, PSO, PHC, and methods based on eigenvalue analysis. However, the best scheme was based on the combination of PSO and PHC. Our major contribution is the utilization of ADE. Therefore, PSO and PHC solve the problem very efficiently (timeliness and performance-based) [27]. Our further research work will make emphasis on the utilization of other algorithms.

3. Supply Chain Case Study

Our case study focuses on inventory and labor concerns in a manufacturing supply chain. The nonlinear SD model of this supply chain is a simplified version of Sterman's original model [9]. It has two submodels: inventory (Figure 3) and labor (Figure 4).

The inventory management sector (Figure 3) is represented by two state variables: inventory and work in process inventory. The variable work in process inventory represents all the stages of the production process, where intermediate inventory is created. The variable Inventory represents the finished goods inventory. This model assumes that orders are filled as they arrive and the ones that cannot be filled immediately are lost as customers seek other sources of supply.

Table 1: Current policy.

Parameter	Value	Unit
Manufacturing cycle time	8	Weeks
Inventory adjustment time	12	Weeks
Average duration of employment	100	Weeks
Average time to fill vacancies	8	Weeks
Labor adjustment time	19	Weeks
Vacancy adjustment time	4	Weeks
WIP adjustment time	6	Weeks
Minimum order processing time	2	Weeks
Safety stock coverage	2	Widgets

The labor sector (Figure 4) is represented by two state variables: vacancies and labor. The stock of vacancies is the supply line or order of workers that have been placed but not yet filled. This states that workers cannot be instantly hired. Hiring takes time: positions must be authorized, and vacancies must be created. The labor force is a stock of people, which is increased by the hiring rate and decreased by the quit rate. This last rate includes voluntary quits and retirements, excluding the possibility of layoffs.

The supply-chain behavior is impacted by interactions between inventory-management policies and labor-adjustment policies. To capture the impact of these policies, the model uses four state variables and several parameters. Two of those parameters—productivity and standard workweek—have constant values of 0.25 widgets/person and 40 hours/week, respectively. The remaining nine parameters (see Table 1) are considered variables that managers can set to design stabilization policies. The current values for these parameters are shown in Table 1. The goal is to find a policy that maintains the inventory and labor state variables at equilibrium and avoids large oscillations in the inventory.

Customer orders are arriving at the rate of 10000 widgets/week. After the system remains in equilibrium for the first five weeks, customer orders experience a linear increment for the next fifty weeks until reaching 20% of its original value, where they remain constant. The resulting behavior of the variables inventory and labor (Figure 5) shows several oscillatory fluctuations. These fluctuations are caused by delays in production.

3.1. Optimization Problem. In order to determine a stabilization policy for these two state variables, we solved the following optimization problem using our proposed global search algorithm (PSO). The algorithm was run at the fifth week using the setting mentioned below:

(1) number of iterations = 150,

(2) swarm size = 30 particles,

(3) neighborhood size = 4,

(4) inertia weight = 0.5,

(5) iteration lag = 5,

(6) cognitive coefficient = 1.2,

(7) social coefficient = 1.2.

Table 2

Parameter	Empirical rule of choice
Swarm size	From 20 to 40 [28]
Inertia weight	In]0, 1[[21]
Cognitive coefficient	Suggestion 1.43 [28]
Social coefficient	Suggestion 1.43 [28]

These parameter values were chosen after doing some initial experiments with the empirical rules selected to guide the choice (see Table 2).

The new parameter set associated to the stabilization policy is shown in Table 3. The objective function (ADE) was improved by 82%. It took 189 seconds to calculate this policy after 150 iterations of the algorithm (the algorithm was executed on a 1.86 GHz Pentium PC with 1 GB of memory).

Let x_1 = inventory, x_2 = labor, x_3 = work in process inventory, and x_4 = vacancies.

The minimization problem considered the first two state variables as the variables of interest. The following weights were assigned $w_1 = 0.6$, $w_2 = 0.4$ to represent the concern of management in the inventory

$$\begin{aligned}
\text{minimize} \quad & \sum_{s=1}^{2}\left\{ w_s \int_0^{200} |x_s(t) - a_s|\,dt \right\}, \\
\text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}), \\
& \mathbf{x}_0^T = \begin{bmatrix} 40000 & 1000 & 80000 & 80 \end{bmatrix}, \\
& 1 \le \text{Manufacturing Cycle Time} \le 50, \\
& 1 \le \text{Inventory Adjustment Time} \le 50, \\
& 50 \le \text{Average Duration of Employment} \le 150, \\
& 1 \le \text{Average Time to Fill Vacancies} \le 50, \\
& 1 \le \text{Labor Adjustment Time} \le 50, \\
& 1 \le \text{Vacancy Adjustment Time} \le 50, \\
& 1 \le \text{WIP Adjustment Time} \le 50, \\
& 1 \le \text{Minimum Order Processing Time} \le 50, \\
& 1 \le \text{Safety Stock Coverage} \le 50, \\
& 10000 \le a_1 \le 150000, \\
& 100 \le a_2 \le 10000.
\end{aligned} \tag{8}$$

Figure 6 shows the behavior of the state variables when the stabilization policy is applied at the fifth week. It clearly shows the convergence of the ADE with minimal oscillation. Two main actions have been taken to respond to the change in customer orders. First, the increment in production has been achieved by decreasing the manufacturing cycle time. Second, by decreasing the time to adjust, labor will help to

Table 3: Parameter values for policy using PSO algorithm.

Parameter	Value	Unit
Manufacturing cycle time	1.98	Weeks
Inventory adjustment time	12.24	Weeks
Average duration of employment	76.67	Weeks
Average time to fill vacancies	1.22	Weeks
Labor adjustment time	2.04	Weeks
Vacancy adjustment time	23.85	Weeks
WIP adjustment time	21.94	Weeks
Minimum order processing time	4.28	Weeks
Safety stock coverage	3.50	Widgets
a_1 (EP for Inventory)	93359	Widgets
a_2 (EP for Labor)	1200	People

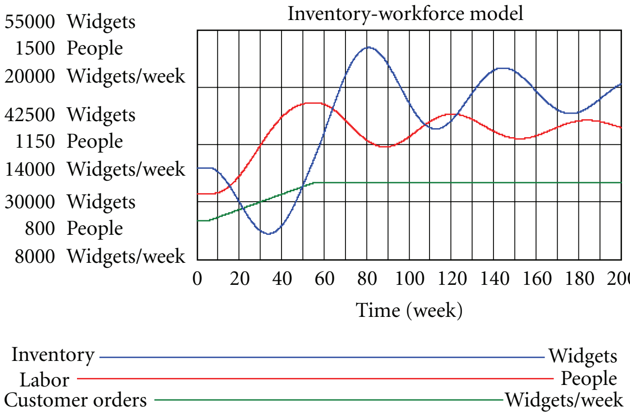

Figure 5: Behavior of state variables for the current policy.

approach production more closely to the desired production rates. As a result, the EP of the inventory variable has increased from 40000 units to 93359 units. The EP for the labor variable remains not far from its original value of 1000 people. Stabilization of both variables has been achieved in approximately 80 weeks.

3.2. Testing for Policy Robustness. The stabilization policy was tested by generating a sudden change in weeks 80 and 100 in the customer orders and showing the system's response to this change. The different percentage changes in customer orders and responses are shown in Table 4. Figure 7 depicts the robust behavior of the inventory and labor variables to the changes. These variables showed a sharp increase or decrease in their levels necessary to adapt to the changes before reaching new equilibrium points. Stability returns approximately in week 130.

3.3. Comparing Polices with a Local Search Algorithm. This methodology does not require to find the global optimum to obtain satisfactory reduction in instability. Using a local search algorithm, we can obtain a quick convergence of the ADE in just few seconds. Although the time to find the

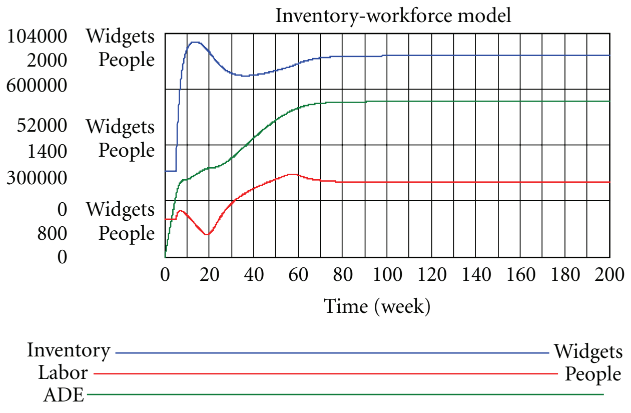

Figure 6: Behavior of state variables with policy that uses PSO algorithm.

optimal solution is an important factor in selecting a search algorithm, the quality of such solution in terms of oscillation reduction has to be analyzed. For that reason, we decided to compare the results obtained by solving the optimization problem using our global search algorithm PSO (Figure 6) and the ones obtained by using the local search algorithm Powell hill climbing (PHC) [29].

We tested two starting EPs for the PHC algorithm. In the first test (PHC1) it was used the original EP of the model (for the inventory and labor variables) as the starting point which was $(a_1, a_2) = (40000, 1000)$. For the second test (PHC2), we used the EP obtained by the PSO algorithm, which was $(a_1, a_2) = (93359, 1200)$. In both tests it took around 15 seconds to find the solution. We can see from Figure 8 that after some fluctuations policy PHC1 achieves stabilization of the inventory and labor variables approximately in 70 weeks. Policy PCH2 stabilizes the system in 120 weeks at a lower EP that the one it started.

Comparing the policies obtained by PCH and PSO, we can see that the second one (PSO) shows less fluctuations before reaching the EP due to the higher inventory level. Both algorithms generate the same EP for the labor variable. However, the policy with PSO gets that by increasing the manpower smoothly after a small trough, while the policies with PCH obtains the stability after several decreasing fluctuations.

4. Conclusions

This paper proposes a methodology, based on the ADE, to eliminate or minimize oscillatory behaviors of the supply chain. Our approach utilizes the modeling flexibilities of system dynamics to model the supply chain and the potential of the PSO algorithm to scan the search space to solve the stabilization problem.

We tested our methodology by increasing and decreasing the customer orders in a manufacturing supply chain. We showed that our approach can stabilize the behavior of the state variables despite these fluctuations. We concluded that after a sudden perturbation of the system the stabilization

Table 4: New EPs reached after the system was perturbed.

Percentage change in customer orders	Sudden change (week)	New EP for inventory (widgets)	New EP for labor (people)
−10%	80	84024	1079
−5%	100	88692	1139
+5%	100	98027	1260
+10%	80	102695	1320

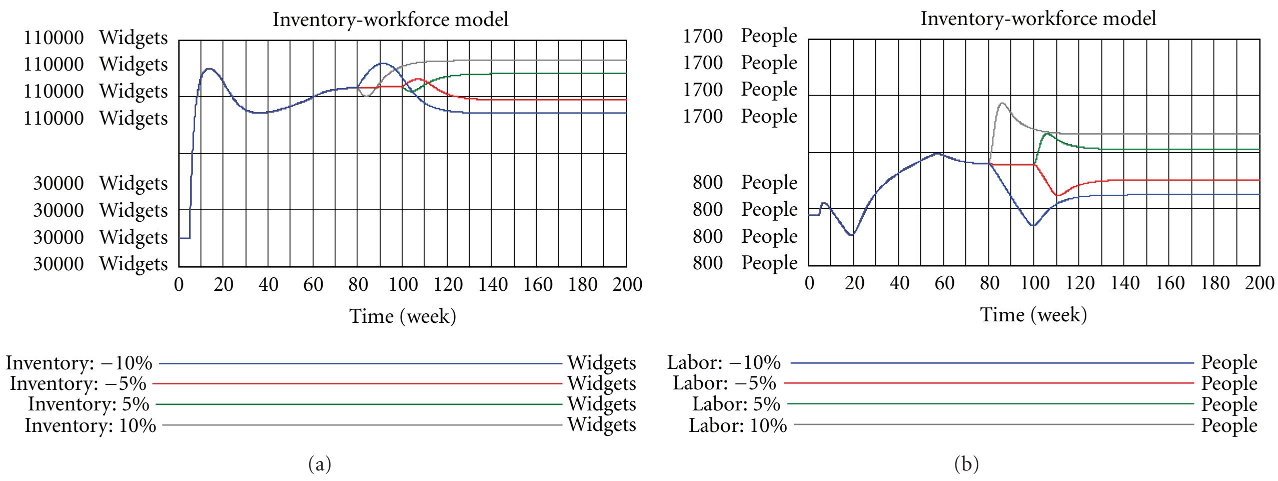

Figure 7: Behavior of state variables after a sudden change in customer orders.

policy remains stable requiring a period of adaptation to the changes before reaching new EPs.

We compared the results of our proposed approach that uses the global search algorithm PSO with the ones obtained by the local search algorithm PHC. Because PSO does not depend on the starting point of the state variables, it provides a more expanded and deeper search of the space to find the EP. We concluded that the PSO algorithm provided a better solution than the PHC algorithm in terms of fewer oscillations. PHC achieved faster stability when started from the original EP of the model.

5. Future Work

We propose to test the performance of this local best PSO algorithm using a real-size model of the supply chain. It will be interesting to compare the results of this algorithm with the ones obtained with other evolutionary algorithms such as GAs and PSO variations and hybrids.

Currently, this methodology is focused in generating stabilization policies at the strategic and tactical levels of the supply chain where SD modeling is more suitable. We propose to extend this research to the operational level by developing a methodology that will propagate stability from the higher to the lower level of the supply chain. We plan to meet this challenge by using hybrid simulation (SD and discrete event simulation) to model the different levels of the supply chain and a hierarchical approach to coordinate between the policies obtained at these levels.

Appendix

Related Definitions and Theorems

Here, we present some important definitions and theorems that provide the support to understand our methodology.

Definition 1. The point $\mathbf{x}^{\text{eq}} \in R^n$ is said to be an equilibrium point of the differential equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))(\dot{\mathbf{x}}(t) = \partial\mathbf{x}(\mathbf{t})/\partial\mathbf{t})$ if it has the property that once the corresponding system reaches $\mathbf{x}^{\text{eq}}$ at time t_{eq}, it will remain at $\mathbf{x}^{\text{eq}}$ for all future time; in other words, $\mathbf{f}(\mathbf{x}(t)) = \mathbf{0}$ for all $t \geq t_{\text{eq}}$.

Definition 2. Consider the system defined by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)); \mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{x}(t) \in R^n$; $\mathbf{f} : R^n \to R^n; \mathbf{x}(t) = [x_s(t)] = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $s = 1, \ldots, n$. The state variable x_s is defined to be stable (around the EP x_s^{eq}) if it is bounded; that is, there is a finite number M_s such that $|x_s(t) - x_s^{\text{eq}}| \leq M_s$ (the symbol $|c|$ represents the absolute value of c). If this condition holds for all state variables, then the system is said to be stable.

Definition 3. Consider the system defined by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$; $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{x}(t) \in R^n$; $\mathbf{f} : R^n \to R^n$; $\mathbf{x}(t) = [x_s(t)] = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $s = 1, \ldots, n$. The state variable x_s is defined to be *asymptotically stable* (around the EP x_s^{eq}) if it is both stable (satisfies Definition 2), and additionally, we have $\text{Lim}_{t\to\infty}(x_s(t) - x_s^{\text{eq}}) \to 0$. If these two conditions hold for all state variables, then the system is said to be *asymptotically stable*.

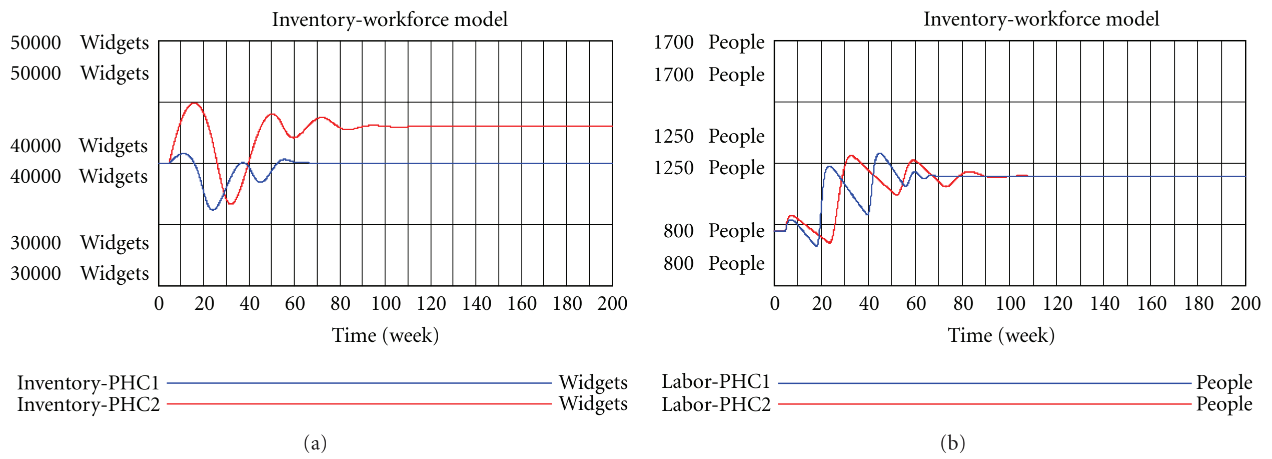

FIGURE 8: Two stabilization policies using PHC algorithm.

Definition 4. Consider the system defined by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$; $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{x}(t) \in R^n; \mathbf{f} : R^n \rightarrow R^n; \mathbf{x}(t) = [x_s(t)] = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $s = 1, \ldots, n$. For the state variable x_s, the accumulated deviations from its EP x_s^{eq} is defined as $\int_0^\infty |x_s(t) - x_s^{\text{eq}}| dt$.

Theorem 1. *Consider the system defined by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$; $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{x}(t) \in R^n$; $\mathbf{f} : R^n \rightarrow R^n$; $\mathbf{x}(t) = [x_s(t)] = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $s = 1, \ldots, n$. The state variable x_s is asymptotically stable (around the EP x_s^{eq}) if $\int_0^\infty |x_s(t) - x_s^{eq}| dt$ converges.*

Theorem 2. *Consider the system defined by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$; $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{x}(t) \in R^n$; $\mathbf{f} : R^n \rightarrow R^n$; $\mathbf{x}(t) = [x_s(t)] = [x_1(t), x_2(t), \ldots, x_n(t)]^T$, $s = 1, \ldots, n$. If $\int_0^\infty |x_s(t) - a_s| dt$ converges, then $a_s = x_s^{eq}$.*

References

[1] B. Bernanke, "Financial Markets, the Economic Outlook, and Monetary Policy," 2008, http://www.federalreserve.gov/newsevents/speech/bernanke20080110a.htm.

[2] J. Lahardt and C. Dougherty, "US retools economy, curbing thirst for oil," *The Wall Street Journal—Business*, vol. A1, p. A18, 2008.

[3] The Economist, The Collapse of Manufacturing, Editorial from the The Economist.com, 2009.

[4] J. Sterman, "Operational and behavioral causes of supply chain instability," in *The Bullwhip Effect in Supply Chains: A Review of Methods, Components and Cases*, O. Carranza and F. Villegas, Eds., Palgrave Macmillan, Basingstoke, UK, 2006.

[5] G. Small and J. Oluyowe, The Significance of Debt, Human Nature and the Nature of Land on Real Estate Cycles, Pacific Rim Real Estate Society International Conference Sydney, 2000.

[6] W. Lingyun, C. Yueting, R. Changrui, and D. Jin, "A research review on dynamic performance analysis of supply chain system," in *Proceedings of the System Modeling and Simulation: Theory and Applications, Asian Simulation Conference*, pp. 163–167, Springer, 2006.

[7] P. Mohapatra, "Nonlinearities in system dynamics models—part II," *Dynamica*, vol. 6, no. 1, pp. 36–52, 1980.

[8] P. McSharry, "Optimisation of system dynamics models using genetic algorithms," submitted to *World Bank Report*.

[9] J. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*, McGraw-Hill, Boston, Mass, USA, 2000.

[10] P. Gonçalves, J. Hines, and J. Sterman, "The impact of endogenous demand on push-pull production system," *System Dynamics Review*, vol. 21, no. 3, pp. 187–216, 2005.

[11] R. Eberlein, "Simplification and understanding of models," *System Dynamics Review*, vol. 5, no. 1, pp. 51–68, 1989.

[12] N. Forrester, "Eigenvalue analysis of dominant feedback loops," in *Proceedings of the International Sys-tem Dynamics Conference*, Albany, NY, USA, 1983.

[13] C. Kampmann, "Feedback loop gains and system behavior," in *Proceedings of the International System Dynamics Conference*, Albany, NY, USA, 1996.

[14] B. Güneralp, "Towards coherent loop dominance analysis: progress in eigenvalue elasticity analysis," in *Proceedings of the International System Dynamics Conference*, Boston, Mass, USA, 2005.

[15] H. S. Khalil, *Nonlinear Systems*, Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[16] S. Lim, *Analysis and control of linear parameter-varying systems*, Ph.D. thesis, Stanford University, Stanford, Calif, USA, 1999.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[18] S. Cui and D. S. Weile, "Application of a novel parallel particle swarm optimization to the design of electromagnetic absorbers," in *Proceedings of the IEEE Antennas and Propagation Society International Symposium and USNC/URSI Meeting*, pp. 41–44, Washington, DC, USA, July 2005.

[19] R. Hassan, B. Cohanim, O. De Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pp. 1138–1150, Austin, Tex, USA, April 2005.

[20] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, John Wiley & Sons, West Sussex, UK, 2005.

[21] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Piscataway, NJ, USA, 1998.

[22] A. Haq and G. Kannan, "Effect of forecasting on the multi-echelon distribution inventory supply chain cost us-ing neural network, genetic algorithm and particle swarm optimization," *International Journal of Services Operations and Informatics*, vol. 1, no. 1-2, pp. 1–22, 2006.

[23] M. Hanwu, Y. Xiang, and Z. Dengfan, "The research into ILRIP for single-stage logistics distribution network under stochastic demand based on JITD," in *Proceedings of the 7th International Conference on Service Systems and Service Management (ICSSSM '10)*, pp. 1–6, Tokyo, Japan, 2010.

[24] Y. Huang, Z. Qiu, and Q. Liu, "Supply chain network design based on fuzzy neural network and PSO," in *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL '08)*, pp. 2189–2193, September 2008.

[25] E. Mehdizadeh and R. Tavakkoli-Moghaddam, "A fuzzy clustering PSO algorithm for supplier base management," *International Journal of Management Science and Engineering Management*, vol. 4, no. 4, pp. 311–320, 2009.

[26] C. Soares, G. Dozier, E. Lodree, J. Phillips, K. Nobles, and W. P. Yong, "Optimization of the multiple retailer supply chain management problem," in *Proceedings of the 46th Annual Southeast Regional Conference*, pp. 490–495, Auburn, Ala, USA, March 2009.

[27] A. Sarmiento, *A methodology for stabilizing the supply chain*, Ph.D. thesis, University of Central Florida, Department of Industrial Engineering and Management Systems, 2009.

[28] M. Clerck, *Particle Swarm Optimization*, STE Ltd, Newport Beach, Calif, USA, 2006.

[29] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, 1964.