

Multi-Echelon Inventory Planning System: Multiple Agent Hybrid Control (MAHCA) Implementation

Wolf Kohn

*Hynomics Corporation,
10632 NE 37th Circle, Build. 23, Kirkland, WA 98033-7921*

Vladimir Brayman

*Hynomics Corporation,
10632 NE 37th Circle, Build. 23, Kirkland, WA 98033-7921*

Jeffrey B. Rummel

*Department of Mathematics and Computer Science,
University of California at San Diego, San Diego, CA 92093*

ABSTRACT: This paper provides a constructive procedure for the implementation of a multiechelon single-product optimal, reactive distributed inventory planner. The paper serves as an illustration of the design procedure for generating models for a network of hybrid agents (Multiple Agent Hybrid Control Architecture, MAHCA) that implement the inventory planner. The objective of the planner is to generate a distributed stocking policy that approximates the optimal centralized stocking policy that minimizes the expected value of the multi-echelon operation cost.

AMS subject Classification: 93A13, 93C15, 93D05

Keywords and Phrases: Hybrid Systems, Multi-echelon Inventory Systems, Software Agents

1 Introduction

This paper provides a step-by step description of a design procedure of a MAHCA network for the deployment of a reactive, agent based planner for a single product, multi-echelon inventory system [1], [2], [9]. The objective of the system is to generate a distributed stocking plan policy. This policy approximates a centralized stocking policy, which minimizes the expected value of a suitably defined cost functional. A precise formulation of the system follows.

Consider a directed graph $G = (V, A)$ which represents the distribution network for a product p . Each vertex j in the set V represents a distribution center (DC).

The existence of an edge (i, j) implies that the DC represented by i acts a source of supply for the DC represented by j . When a DC can receive supply from multiple sources, it can be assumed that there is a prespecified order in which the sources will be polled for supply. The lead-time between DCs is one time period. There is one infinite supply source that supplies to two nodes in the network. Demands for item p at vertex v in time period t follow a distribution whose c.d.f is given by $F_i^t(\cdot)$. Demand is assumed to be greater than zero in every period. The demands of any two nodes are assumed to be independent.

The cost of holding inventory at i per period is h_i . The penalty for not satisfying demand is s_i per unit of demand. The transportation lead-time from the infinite source external supplier to each node j directly connected to it is given by $l_j > 1$ time periods and is deterministic.

The paper is organized in 4 sections: Section 1 is this introduction. Section 2, Model Building, provides a mathematical model for the multi-echelon system. The model is expressed as a collection of sub-models each associated with a distribution center. Section 3, Model Continualization, describes the process of continualization as applied to the sub-models of Section 2. Section 4, Hybrid Dynamic Programming, describes the Hybrid Dynamic Algorithm that implements the MAHCA distributed optimization paradigm. We also provide 2 appendixes: Appendix A contains the formulation of the multi-echelon system and appendix B provides a brief operational description of MAHCA.

2 Model Building

In this section, we describe the formal construction of a dynamic model for each distribution center and a model for their interaction. The step we are describing here is referred to in our technology as the *model building phase* [3], [4]. This activity involves the formulation of flow, rules for material uncertainty and optimality conservation, and rules characterizing flow and material constraints. These rules are used to construct an iterative model representing the dynamics of the relevant supply chain or network. For the purposes of future reference, we call this model the *original model*.

For the purposes of notational commonality and understanding, we have formulated the original model as a collection of sub-models, each corresponding to the dynamics and criteria at each distribution center. This formulation is an input to our continualization process, which, as will be discussed in the next section, generates the domain dynamics model and synchronization constraints used by the decision agents of MAHCA [6].

The connection graph of an example of the proposed one-product multi-echelon system is shown in the Figure 1 below.

Table 1 below lists the model variables for a generic distribution center termed the i -th distribution center; where i is the label of the node representing the center in the graph of the system. The problem includes three types of distribution centers: An infinite source DC ($i = 0$), input DC's ($i = 1, 2$) and regular DC's ($i = 3, 4, 5, 6$).

No	Name	Notation	Description
1	Inventory (State variable)	$x_i(t)$	$x_i(t)$ = the number of units of the product carried over from the previous period at node DC i at the beginning of period t . We assume $x_i(t) \geq 0 \forall i, t$.
2	Ordered amount (Decision variable)	$u_{ij}(t)$	$u_{ij}(t)$ = the number of units of the product ordered by DC i from node $j \in U_i$ at the end of period t where U_i is the index set of the upstream nodes adjacent to node DC i . We assume $u_{ij}(t) \geq 0 \forall i, j, t$.
3	Sent amount (Decision variable)	$v_{ij}(t)$	$v_{ij}(t)$ = the number of units of the product sent from node i to node $j \in V_i$ at the end of period t , where V_i is the index set of the downstream nodes adjacent to node DC i . We assume $0 \leq v_{ij}(t) \leq u_{ji}(t), \forall i, t \forall j \in V_i$.
4	External demand satisfaction (Auxiliary variable)	$y_i(t)$	$y_i(t)$ = the number of units of product delivered by DC i to meet the external demand $d_i(t)$. We assume $0 \leq y_i(t) \leq d_i(t) \forall i, t$.
5	Total demand (Auxiliary variable)	$D_i(t)$	$D_i(t) = d_i(t) + \sum_{j \in V_j} u_{ji}(t-1), \forall i$
6	Total availability (Auxiliary variable)	$S_i(t)$	$S_i(t) = x_i(t) + \sum_{j \in U_i} v_{ji}(t - \omega_{ji}), \forall i > 0$. $S_0(t) = \infty$. Here ω_{ij} is the transportation lead-time from DC j to DC i . Thus for $j > 0, i = 3, \dots, 6, \omega_{ji} = 1$ and for $j = 0, i = 1, 2, \omega_{ji} = I_{ji}$
7	Total amount sent (Auxiliary variable)	$M_i(t)$	$M_i(t) = y_i(t) + \sum_{j \in V_i} v_{ij}(t)$ $M_i(t) \leq S_i(t) \forall i$.

Table 1: Problem Variables

For each DC, the decision variables are listed in Table 2 below. These variables are the ordering and sending decision variables, external (demand, a stochastic process with known distribution), the storage variable (which is deterministic a posteriori but is a stochastic process in predictive mode). For notational simplicity we will use the same symbol for both forms.

No	Name	Notation	Description
1	External demand (Stochastic, given)	$D_i(t)$	RV with c.d.f. $F_i^t(\cdot)$.
2	Inventory (State variable)	$x_i(t)$	$x_i(t)$ = the number of units of the product carried over from the previous period at node DC i at the beginning of period t . We assume $x_i(t) \geq 0 \forall i, t$.
3	Ordered amount (Decision variable)	$u_{ij}(t)$	$u_{ij}(t)$ = the number of units of the product ordered by DC i from node $j \in U_i$ at the end of period t where U_i is the index set of the upstream nodes adjacent to node DC i . We assume $u_{ij}(t) \geq 0 \forall i, j, t$.
4	Sent amount (Decision variable)	$v_{ij}(t)$	$v_{ij}(t)$ = the number of units of the product sent from node i to node $j \in V_i$ at the end of period t , where V_i is the index set of the downstream nodes adjacent to node DC i . We assume $0 \leq v_{ij}(t) \leq u_{ji}(t), \forall i, t \forall j \in V_i$.

Table 2: Decision Variables

For each DC the criterion is given by:

$$\min_{\substack{u_{ij}, j \in U_i \\ v_{ij}, j \in V_i}} E \left(\sum_{t=0,1,\dots} [s_i \max\{0, d_i(t) - y_i(t)\} + h_i x_i(t)] \right) \quad (1)$$

where

$$y_i(t) = \min(d_i(t), S_i(t) - \sum_{j \in V_i} v_{ij}(t))$$

Here E is the expectation operator defined with respect to the corresponding demand distribution. We expect the horizon of the problem to be some kind of moving window but we have not discussed this yet.

We note that the criterion for the centralized version of the problem in which the criterion is the sum over i of all the criteria of the individual DC's. Figure 1 illustrates the supply chain associated with the given system. The solid lines represent requests, the dotted lines represent deliveries. The state of the chain is the vector of excess demand variables.

Request flow ---
 Delivery flow —

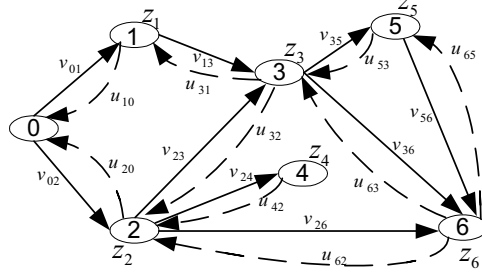


Figure 1: Multi-Echelon System

The Storage dynamics for each distribution center is given by

$$x_i(t+1) = S_i(t) - M_i(t). \quad (2)$$

We define *excess demand* as the difference between total demand and availability. That is,

$$z_i(t) = D_i(t) - S_i(t) = d_i(t) + \sum_{j \in V_i} u_{ji}(t-1) - x_i(t) - \sum_{j \in U_i} v_{ji}(t-1). \quad (3)$$

Now, using the equation for $y_i(t)$, the criterion is given by

$$\min_{\substack{u_{ij}, j \in U_i \\ v_{ij}, j \in V_i}} E \left(\sum_{t=0,1,\dots} s_i \max\{0, d_i(t) - (S_i(t) - \sum_{j \in V_i} v_{ij}(t))\} + h_i x_i(t) \right) \quad (4)$$

We can rewrite (4) in terms of $z_i(t)$ as follows.

$$\begin{aligned} \min_{\substack{u_{ij}, j \in U_i \\ v_{ij}, j \in V_i}} E \left(\sum_{t=0,1,\dots} s_i \max\{0, z_i(t) - \sum_{j \in V_i} (v_{ij}(t) - u_{ji}(t))\} \right. \\ \left. + h_i [d_i(t) - (z_i(t) + \sum_{j \in V_i} (v_{ij}(t-1) - u_{ji}(t-1)))] \right) \end{aligned} \quad (5)$$

The storage dynamics then becomes

$$\begin{aligned}
x_i(t+1) &= D_i(t) - z_i(t) - M_i(t) \\
&= d_i(t) - y_i(t) - [z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)] \\
&= d_i(t) - \min\{S_i(t) - \sum_{j \in V_i} v_{ij}(t), d_i(t)\} - [z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)] \\
&= \max\{0, z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)\} - [z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)] \\
&= \max\{0, -[z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)]\}. \tag{6}
\end{aligned}$$

Thus from (6) and (3), we obtain the following recursion for the excess demand dynamics.

$$\begin{aligned}
z_i(t+1) &= d_i(t+1) + \sum_{j \in V_i} u_{ji}(t) - \sum_{j \in U_i} v_{ji}(t - \omega_{ji} + 1) - \\
&\quad \max\{0, -[z_i(t) + \sum_{j \in V_i} v_{ij}(t) - u_{ji}(t-1)]\}. \tag{7}
\end{aligned}$$

Now we are going to define the consolidated discrete dynamic model for each DC_{*i*}. This model constitutes the input data for our continualization procedure that we will describe in the next section. First we define an auxiliary variable $\xi_i(t)$, the planning decision vector $w_i(t)$, input vector $g_i(t)$, and maximum supply delay τ_i associated with DC_{*i*}.

Suppose that $V_i = \{j_1 < \dots < j_{m_i}\}$, then

$$w_i(t) = \begin{bmatrix} w_i^1(t) \\ \vdots \\ w_i^{m_i}(t) \end{bmatrix} = \begin{bmatrix} v_{ij_1}(t) \\ \vdots \\ v_{ij_{m_i}}(t) \end{bmatrix} \tag{8}$$

$$g_i(t - \tau_i) = \begin{bmatrix} \sum_{j \in V_i} u_{ji}(t - \tau_i - 1) \\ \sum_{j \in V_i} u_{ji}(t - \tau_i) \\ \sum_{j \in U_i} v_{ij}(t - \tau_i - \omega_{ij}) \end{bmatrix} \tag{9}$$

where

$$\tau_i = \max_{j \in U_i} \{\tau_j + \omega_{ji}\} \tag{10}$$

and $\tau_0 = 0$. Then the dynamics is given by the follow stochastic iteration.

$$z_i(t+1) = SAT(\xi_i(t))\xi_i(t) + C_2 g_i(t - \tau_i) + d_i(t) \tag{11}$$

where

$$SAT(f) = \begin{cases} 1 & \text{if } f > 0 \\ 0 & \text{if } f \leq 0 \end{cases} \tag{12}$$

and

$$\xi_i(t) = -z_i(t) + B^i w_i(t - \tau_i) + C_1 g_i(t - \tau_i)$$

with $B^i = [1, \dots, 1]$ is a row vector of length m_i , $C_1 = [1, 0, 0]$, and $C_2 = [0, 1, 1]$.

The delay τ_i introduced in the dynamic equation of DC $_i$ (11) provides a coordination mechanism between the actual flow of the product and the state of the distribution center. This is not the only possible mechanism, but it is a conservative one. It can be easily shown that this mechanism guarantees that real time policies generated from our agent procedure are feasible in the mean sense relative to the probability space induced by the demand on the state space. This mechanism, or one like it, is necessary if the MAHCA system is to be implemented as a real time planner. As we shall see below, in order to ensure inter-agent synchronization, we need additional dynamics.

Since we have formulated optimization problem P_i solved by the agent planning for DC $_i$ to independent of the optimization problems solved by the agents planning for the other DC $_j$'s in the system, we need to establish constraints which will ensure that P_i 's solution is compatible with the solutions of the other P_j 's. Our MAHCA technology provides two ways to address this synchronization problem: (1) Real-time synchronization and (2) Rule based synchronization. In real-time synchronization, sensory data is used to establish whether the current plan behavior coincides with the model behavior. We will briefly discuss real-time synchronization in section 4. We shall now describe a model for rule-based synchronization associated with the multi-echelon system.

For each DC $_i$, we define by the following iteration.

$$\eta_i(t+1) = \eta_i(t) + \sum_{j \in U_i} (u_{ij}(t-1-\tau_i) - v_{ji}(t-\tau_i)) \quad (13)$$

where $\eta_i(0) = 0$.

We note that $\eta_i(t)$ measures the cumulative discrepancy between what a DC $_i$ orders and what it receives. For optimal performance of the system, we need $\sum_t E(\eta_i(t)) = 0$ for all i because if $\sum_t E(\eta_i(t)) > 0$, then the optimal policy for DC $_i$ introduces a negative bias on the policies of the upstream DC $_j$'s. On the other hand $\sum_t E(\eta_i(t)) < 0$ leads to unbounded behavior. Thus, the synchronization rule is implemented by incorporating (13) into the state equations of the model and by adding $\eta_i(t)$ to the criterion.

Next we derive an equivalent iterative representation of the cost functional. First, rewrite the expression in (2) inside the expectation operator in terms of $\xi_i(t)$ taking care of the first and last terms in the summation. Then when we add the term $\eta_i(t)$ to the summand, we obtain the following expression.

$$\sum_t \{s_i(-\xi_i(t) + SAT(\xi_i(t))\xi_i(t)) + h_i[d_i(t) + \xi_i(t)] + \eta_i(t)\}.$$

Next we define a summing variable, $J_i(t)$ as follows.

$$J_i(t+1) = J_i(t) + (h_i - s_i + SAT(\xi_i(t))\xi_i(t) + h_i d_i(t) + \eta_i(t)) \quad (14)$$

with initial condition $J_i(0) = J_{i_0}$.

The criterion for the DC_{*i*} then simply becomes

$$\min_{w_i, u_{ji}: j \in U_i} E\{J_i(N)\}$$

where N is the time horizon of the plan. Thus the optimization problem characterizing the policies of each DC_{*i*} is given by

$$\min_{w_i, u_{ji}: j \in U_i} E\{J_i(N)\} \quad (15)$$

subject to the dynamic constraints (11), (13) and (14) and their corresponding initial conditions plus the structural constraints that $\forall t, u_{ij}(t) \geq 0, v_{ij}(t) \geq 0$ and $\xi_i(t) \geq 0$.

This is the original problem for each DC_{*i*}. We note that the formulation is characterized by a set of iterations constraining a terminal optimization problem. This is always the result of our model building operation. Notice also that once the conservation principles characterizing the chain are given, the procedure is almost completely mechanical.

We conclude this section with a flow chart (Figure 2) that summarizes the model building step of our procedure.

3 Continualized Model

In this section we develop a continualized approximation of the model of Section 2. Figure 3 shows the main elements of the agent that generates a planning policy for the DC_{*i*}. The outputs of the continualization procedure are differential models [7], [8]. For the application in this paper, these are the continualized versions of the DC_{*i*} dynamics model (11), the criterion model (14), the controller model and the synchronization constraint (13). For the purposes of simplifying the manipulations that follow, we will rewrite these three models in predictive form. We will also combine the three models into a single vector iteration.

For each DC_{*i*}, let $r_i(t) = [z_i(t), \eta_i(t), J_i(t)]^T$. Then, the predictive-mode model is obtained as follows. First suppose that $V_i = \{j_1 < \dots < j_{m_i}\}$, then

$$\tilde{u}_i(t) = \begin{bmatrix} \tilde{u}_i^1(t) \\ \vdots \\ \tilde{u}_i^{m_i}(t) \end{bmatrix} = \begin{bmatrix} u_{j_1 i}(t) \\ \vdots \\ u_{j_{m_i} i}(t) \end{bmatrix}$$

From (11), we can then obtain an expression for the demand in terms of the state and total delivery and request decision variables for DC_{*i*}, $r_i(t)$, $w_i(t)$ and \tilde{u}_i plus $g_i(t)$ and the request and delivery vector for the down stream DC_{*j*}'s. We construct the predictive state equations of DC_{*i*} by simply advancing the time variable by τ_i . Then we replace in the resulting vector iteration all the terms that include $d_i(t)$ by its expression in terms of the state and control variables. The resulting state equation

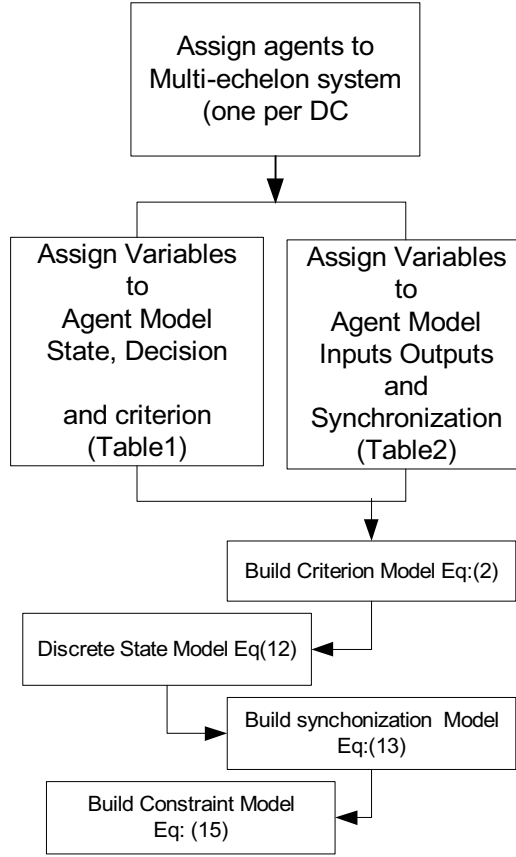


Figure 2: Model Building

will be of the form

$$r_i(t + \tau_i + 1) = \sum_{j=0}^{\tau_i} \tilde{A}_{ij}(t) \cdot r_i(t + j) + \tilde{B}_i(t) \cdot w_i(t) - \tilde{C}_i(t) \cdot \tilde{u}_i(t - 1) + \tilde{G}_i(t - 1) \cdot g_i(t - 1) + \sum_{s=0}^{\tau_i+1} \tilde{D}_{is}(t + s) \cdot d_i(t + s) \quad (16)$$

where the coefficient matrices are

$$\tilde{\mathbf{A}}_i(t) = \begin{bmatrix} -SAT(\xi_i(t)) & 0 & 0 \\ 0 & 1 & 0 \\ -h_i + s_i - SAT(\xi_i(t)) & 1 & 1 \end{bmatrix}, \tilde{\mathbf{B}}_i(t) = \begin{bmatrix} -SAT(\xi_i(t)) \cdot B^i \\ 0 \\ (-h_i + s_i - SAT(\xi_i(t))) \cdot B^i \end{bmatrix}$$

$$\tilde{\mathbf{C}}_i(t) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tilde{\mathbf{G}}_i(t) = \begin{bmatrix} -SAT(\xi_i(t)) \cdot C_1^i + C_2^i \\ -C_3^i \\ (-h_i + s_i - SAT(\xi_i(t))) \cdot C_1^i \end{bmatrix},$$

$$\tilde{\mathbf{D}}_{i0}(t) = \begin{bmatrix} 0 \\ 0 \\ h_i \end{bmatrix}, \tilde{D}_{i1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

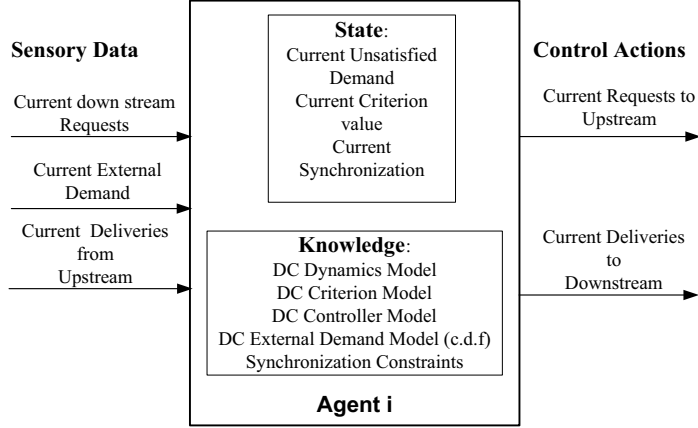


Figure 3: Agent Planner For DC_i

The next step in the continualization process is to construct a feasible symbolic form of decision policy as a function of the state and demand. Because of the quasi-linearity of (16) and of the conservation principles in supply chain problems, we can express the generic feasible policy as a quasi-linear feedback of the form

$$\begin{aligned}
 w_i(t) &= \sum_{k=0}^{\tau_i-1} \Theta_{ik}^{wr} \cdot \hat{r}_i(t+k) + \sum_{s=0}^{\tau_i} \Theta_{is}^{wd} \cdot \hat{d}_i(t+s) + \Theta_i^{wg} \cdot \hat{g}_i(t+1) \\
 \tilde{u}_i(t) &= \sum_{k=0}^{\tau_i-1} \Theta_{ik}^{ur} \cdot \hat{r}_i(t+k) + \sum_{s=0}^{\tau_i} \Theta_{is}^{ud} \cdot \hat{d}_i(t+s) + \Theta_i^{ug} \cdot \hat{g}_i(t+1). \quad (17)
 \end{aligned}$$

In (17), the $\hat{\cdot}$ on top of a variable denotes expectation conditioned on the data up to time t . Similarly, from (16) we obtain

$$\begin{aligned}
 \hat{r}(t+\tau_i+1) &= \sum_{j=0}^{\tau_i} \tilde{A}_{ij}(t) \cdot \hat{r}_i(t+j) + \tilde{B}_i(t) \cdot w_i(t) - \tilde{C}(t) \cdot \tilde{u}_i(t-1) \\
 &\quad + \tilde{G}_i(t-1) \cdot g_i(t-1) \\
 &\quad + \sum_{s=0}^{\tau_i+1} \tilde{D}_{is} \cdot \int_0^\infty d_i \cdot dF_i^{t+s}(d_i) \quad (18)
 \end{aligned}$$

If we then use the optimum coefficient functions in (17) and replace the resulting expressions for the decision variables $w_i(t)$ and $\tilde{u}_i(t)$ in (18), we will obtain an expression for the optimum state trajectory for the DC_i in terms of the external demand and the request and delivery trajectories. Specifically, in (17), the ‘coefficient’ functions are to be determined so as to satisfy the optimality in (15), namely,

$$\min_{\Theta_{ik}^{lm}} E\{r_i(N - \tau_i)\} \quad (19)$$

subject to (17) and (18) and their corresponding initial conditions plus the structural constraints that $u_{ji}(t) \geq 0$, $v_{ij}(t) \geq 0$, and $\xi_i(t) \geq 0$ for all t .

Before we continue to the next step of our continualization procedure, we pause to establish a technical point of importance. We note that the coefficient functions in both (17) and (18) are piecewise twice differentiable (C^2) functions with only isolated point discontinuities that are caused by the *SAT* function. A version of the Smoothing Lemma establishes the existence of twice-differentiable models arbitrarily ‘close’ to a given piecewise C^2 model with finitely many isolated discontinuities where close here means in the strong norm sense. Therefore, we can assume that (16), (17) are the C^2 versions of the model. When continualization is mechanized in a computer procedure, it must be provided with a robust heuristic to ‘round-up’ these point discontinuities.

Plugging (17) into (18), we obtain an expression of predicted behavior dynamics for DC $_i$. For simplicity of the calculations that follow we will express this equation in ‘state space notation’ as

$$\hat{\mathbf{X}}^i(t+1) = \mathbf{F}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{X}}^i(t) + \mathbf{H}^i(t, \Theta^i(t)) \cdot \mathbf{G}^i(t) + \mathbf{E}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{D}}^i(t) \quad (20)$$

where $\hat{\mathbf{X}}^i(t) = [\hat{r}_i(t), \dots, \hat{r}_i(t + \tau_i)]^T$, $\mathbf{G}^i(t)$ is the vector of requests and deliveries to DC $_i$, and $\hat{\mathbf{D}}^i(t) = [\hat{d}_i(t), \dots, \hat{d}_i(t + \tau_i)]^T$ is the vector of external demand forecasts between the current time t and the time $t + \tau_i$.

The state evolution of each DC $_i$ is a Markov process driven by external demand and the decision policy (17). Equation (20) shows how the conditional expectation of this process evolves in time.

The next step in the continualization process involves writing the necessary conditions of optimality in problem (19) for the coefficient functions as a time iteration. For this problem, this is particularly easy because the Hamiltonian is of the form

$$\begin{aligned} \Omega^i(t, \Theta^i(t), \mathbf{X}^i(t), \mathbf{P}^i(t+1)) = \\ [\mathbf{P}^i(t+1)]^T \cdot [\mathbf{F}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{X}}^i(t) + \mathbf{H}^i(t, \Theta^i(t)) \cdot \mathbf{G}^i(t) + \mathbf{E}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{D}}^i(t)] \end{aligned} \quad (21)$$

where $\hat{\mathbf{P}}^i(t) = [\hat{r}_i(t), \dots, \hat{r}_i(t + \tau_i)]$, is a vector multiplier. Thus the optimality condition takes the form

$$\begin{aligned} \frac{\partial \Omega^i(t, \Theta^i(t), \mathbf{X}^i(t), \mathbf{P}^i(t+1))}{\partial \Theta^i} = \\ [\mathbf{P}^i(t+1)]^T \cdot \frac{\partial [\mathbf{F}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{X}}^i(t) + \mathbf{H}^i(t, \Theta^i(t)) \cdot \mathbf{G}^i(t) + \mathbf{E}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{D}}^i(t)]}{\partial \Theta^i} = 0 \end{aligned} \quad (22)$$

For simplicity, let us define the function $\Xi(\Theta^i(t), \hat{\mathbf{X}}^i(t), \mathbf{G}^i(t), \hat{\mathbf{D}}^i(t))$ as follows.

$$\begin{aligned} \Xi(\Theta^i(t), \hat{\mathbf{X}}^i(t), \mathbf{G}^i(t), \hat{\mathbf{D}}^i(t)) = \\ \frac{\partial [\mathbf{F}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{X}}^i(t) + \mathbf{H}^i(t, \Theta^i(t)) \cdot \mathbf{G}^i(t) + \mathbf{E}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{D}}^i(t)]}{\partial \Theta^i}. \end{aligned} \quad (23)$$

Then the Stochastic Approximation (SA) iteration for computing an approximating sequence that satisfies (22) is given by

$$\Theta^i(t+1) = \Theta^i(t) + \epsilon^i(t) \cdot \Xi(\Theta^i(t), \mathbf{X}^i(t), \mathbf{G}^i(t), \hat{\mathbf{D}}^i(t)) \quad (24)$$

where $\epsilon^i(t)$ is a time convergence parameter sequence such that

$$\epsilon^i(t) \rightarrow 0 \text{ as } t \rightarrow T, \sum_t \epsilon^i(t) \rightarrow M^i \text{ (large)}$$

We note that in the SA iteration (24), the convergence scalar parameter has replaced the multiplier function \mathbf{P}^i in (22) and hence no backward recursion is needed. Also, the convergence of this recursion can be established from the smoothness of the construction in (20).

Now equation (20) can also be written in the standard SA form as follows.

$$\begin{aligned} \hat{\mathbf{X}}^i(t+1) - \hat{\mathbf{X}}^i(t) &= (\mathbf{F}^i(t, \Theta^i(t)) - I) \cdot \hat{\mathbf{X}}^i(t) + \mathbf{H}^i(t, \Theta^i(t)) \cdot \mathbf{G}^i(t) + \\ &\quad \mathbf{E}^i(t, \Theta^i(t)) \cdot \hat{\mathbf{D}}^i(t). \end{aligned} \quad (25)$$

Next we continualize (24) and (25). Let $\tilde{\Theta}^i(\sigma)$ and $\tilde{\mathbf{X}}^i(\sigma)$ be continuous processes in the continuous 'time' variable σ which are defined as follows.

$$\begin{aligned} \tilde{\Theta}^i(\sigma) &= \Xi_i(\tilde{\Theta}^i(\sigma), \tilde{\mathbf{X}}^i(\sigma), \tilde{\mathbf{G}}^i(\sigma), \tilde{\mathbf{D}}^i(\sigma)) + \mathbf{B}_0^i(\sigma) \text{ and} \\ \tilde{\mathbf{X}}^i(\sigma) &= \mathbf{F}^i(\sigma, \tilde{\Theta}^i(\sigma) - I) \cdot \tilde{\mathbf{x}}^i(\sigma) + \mathbf{H}^i(\sigma, \tilde{\Theta}^i(\sigma)) \cdot \tilde{\mathbf{G}}^i(\sigma) + \\ &\quad \mathbf{E}^i(\sigma, \tilde{\Theta}^i(\sigma)) \cdot \tilde{\mathbf{D}}^i(\sigma) + \mathbf{B}_1^i \end{aligned} \quad (26)$$

The vector functions, $\mathbf{B}_0^i, \mathbf{B}_1^i$, are convergence controls that tend to zero as $\sigma \rightarrow N$. The 'continuous-time' vector variables in (26) are related to the 'discrete-time' by the following linear interpolation schema.

$$\begin{aligned} \tilde{\Theta}^i(t) &= \Theta^i(t) \quad \forall t \\ \tilde{\Theta}^i(\sigma) &= \frac{(t+1-\sigma)}{\epsilon(t)} \cdot \Theta^i(t) + \frac{(\sigma-t)}{\epsilon(t)} \cdot \Theta^i(t+1) \quad \forall \sigma \in (t, t+1) \end{aligned} \quad (27)$$

$$\begin{aligned} \tilde{\mathbf{X}}^i(t) &= \hat{\mathbf{X}}^i(t) \quad \forall t \\ \tilde{\mathbf{X}}^i(\sigma) &= \frac{(t+1-\sigma)}{\epsilon(t)} \cdot \hat{\mathbf{X}}^i(t) + \frac{(\sigma-t)}{\epsilon(t)} \cdot \hat{\mathbf{X}}^i(t+1) \quad \forall \sigma \in (t, t+1) \end{aligned}$$

Similar formulas hold for $\tilde{\mathbf{D}}^i(\sigma)$ and $\tilde{\mathbf{G}}^i(\sigma)$. The following continuous time optimization problem has a solution that approximates the solution (19) up to order $o(\epsilon^2(N))$.

$$\min_{\theta_{ik}^{lm}} \tilde{\mathbf{X}}_n^i(N - \tau_i) \quad (28)$$

subject to the constrains (27), and their corresponding initial conditions plus the usual positivity constraints.

The last step of our continualization procedure is to transform problem (28) into a convex variational problem that generates the same state and parameter trajectory as (28). We note that the proof of the stated distance between solutions to problem

(28) and the discrete solutions to problem (19) is not difficult. It is a rather lengthy and technical exercise that involves comparing Euler discretization sequences of (27) with corresponding iterations of (24) and (25) and an application of the Arzela-Ascoli theorem for uniform convergence.

Let $\mathbf{Y}^i(\sigma) = [\tilde{\Theta}^i(\sigma), \tilde{\mathbf{X}}^i(\sigma)]$ and $\mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma)$ be the following function.

$$\mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma) = \begin{bmatrix} \Xi_i(\tilde{\Theta}^i(\sigma), \tilde{\mathbf{X}}^i(\sigma), \tilde{\mathbf{G}}^i(\sigma), \tilde{\mathbf{D}}^i(\sigma)) + \mathbf{B}_0^i(\sigma) \\ \mathbf{F}^i(\sigma, \tilde{\Theta}^i(\sigma)_I) \cdot \tilde{x}^i(\sigma) + \mathbf{H}^i(t, \tilde{\Theta}^i(\sigma)) \cdot \tilde{\mathbf{G}}^i(\sigma) \\ + \mathbf{E}^i(\sigma, \tilde{\Theta}^i(\sigma)) \cdot \tilde{\mathbf{D}}^i(\sigma) + \mathbf{B}_1^i \end{bmatrix} \quad (29)$$

Then we can write (26) as the following vector field equation.

$$\dot{\mathbf{Y}}^i(\sigma) = \mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma). \quad (30)$$

By our construction via the Smoothing Lemma $\mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma)$ is twice continuously differentiable in both arguments and hence we can differentiate both sides of (30) with respect to σ to obtain the following spray equation associated with (30).

$$\ddot{\mathbf{Y}}^i(\sigma) = \frac{\partial \mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma)}{\partial \mathbf{Y}^i} \cdot \dot{\mathbf{Y}}^i(\sigma) + \frac{\partial \mathbf{f}^i(\mathbf{Y}^i(\sigma), \sigma)}{\partial \sigma} = \mathbf{F}^i(\mathbf{Y}^i(\sigma), \dot{\mathbf{Y}}^i(\sigma), \sigma). \quad (31)$$

We want to find a function $L^i(\mathbf{Y}^i(\sigma), \dot{\mathbf{Y}}^i(\sigma), \sigma)$ such that a solution of the problem

$$\min_{\mathbf{Y}^i} \int_0^T L^i(\mathbf{Y}^i(\xi), \dot{\mathbf{Y}}^i(\xi), \xi) \cdot d\xi + Y_{2n}^i(T) \quad (32)$$

subject our initial conditions and positivity constraints is also a solution of problem (28). Specifically, we seek a function L^i such that (31) are necessary conditions for optimality in problem (32). In our previous papers, we refer to this computation as the *inverse Lagrangian procedure*. Now the necessary conditions for optimality of (32) are given by the Euler-Lagrange Equations associated with L^i , namely,

$$\frac{d}{d\sigma} \frac{\partial L^i(\mathbf{Y}^i(\sigma), \dot{\mathbf{Y}}^i(\sigma), \sigma)}{\partial \mathbf{Y}_j^i} - \frac{\partial L^i(\mathbf{Y}^i(\sigma), \dot{\mathbf{Y}}^i(\sigma), \sigma)}{\partial \mathbf{Y}_j^i} = 0 \text{ for } j = 1, \dots, n_i. \quad (33)$$

Using (31) in (33) we can write (33) as follows.

$$\sum_{j=1}^{n_i} [L_{\dot{\mathbf{Y}}_m^i \mathbf{Y}_j^i}^i \cdot \dot{\mathbf{Y}}_j^i + L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_j^i}^i \cdot \mathbf{F}_j^i] + L_{\dot{\mathbf{Y}}_m^i \sigma}^i - L_{\mathbf{Y}_m^i}^i = 0 \text{ for } m = 1, \dots, n_i. \quad (34)$$

In (34), we have used the sub-index notation for partial derivatives. By our construction, L is a smooth function. Therefore in (34) we may differentiate with respect to $\dot{\mathbf{Y}}_k^i$. After some algebra, we obtain the following from (34). For all $m = 1, \dots, n_i$ and $k = 1, \dots, n_i$,

$$\begin{aligned} & \sum_{j=1}^{n_i} 2L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_k^i \mathbf{Y}_j^i}^i \cdot \dot{\mathbf{Y}}_j^i + \sum_{j=1}^{n_i} 2L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_k^i \dot{\mathbf{Y}}_j^i}^i \cdot \mathbf{F}_j^i + \sum_{j=1}^{n_i} 2L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_j^i}^i \cdot \mathbf{F}_{j \mathbf{Y}_k^i}^i \\ & + \sum_{j=1}^{n_i} 2L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_k^i \mathbf{Y}_j^i}^i \cdot \mathbf{F}_{j \mathbf{Y}_m^i}^i + 2L_{\dot{\mathbf{Y}}_m^i \dot{\mathbf{Y}}_k^i \sigma}^i \end{aligned} \quad (35)$$

Now we define the symmetric matrix function $\Psi^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i, \sigma)$ to be the Hessian of L^i . That is,

$$\Psi_{km}^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i, \sigma) = L_{\dot{\mathbf{Y}}_k^i \dot{\mathbf{Y}}_m^i}^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i, \sigma) \quad (36)$$

We also have that the total derivative of $\Psi^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i, \sigma)$ with respect to σ is given by

$$\frac{d\Psi^i}{d\sigma} = \Psi_{\sigma}^i + \Psi_{\mathbf{Y}^i}^i \cdot \mathbf{Y}^i + \Psi_{\dot{\mathbf{Y}}^i}^i \cdot \dot{\mathbf{F}}^i. \quad (37)$$

For the Hessian $\left[L_{\dot{\mathbf{Y}}_k^i \dot{\mathbf{Y}}_m^i}^i \right]$ of L^i the equations in (35) define a quasi-linear hyperbolic partial differential equation. Indeed, using (37) and (36) in (35), we obtain the following differential equation for Ψ^i along the characteristics associated with the equation.

$$\frac{d\Psi^i}{d\sigma} = -\frac{1}{2}\Psi^i \cdot (\mathbf{F}_{\dot{\mathbf{Y}}^i}^i)^T - \frac{1}{2}\mathbf{F}_{\dot{\mathbf{Y}}^i}^i \cdot \Psi^i. \quad (38)$$

Note that (38) is just a linear Lyapunov equation!

A positive, convex Lagrangian L_0^i corresponding to (38) is given by the following quadrature.

$$L_0^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i, \sigma) = \frac{1}{2}(\dot{\mathbf{Y}}^i)^T \cdot \Psi^i \cdot \dot{\mathbf{Y}}^i \quad (39)$$

From this Lagrangian an equivalent Lagrangian L_1^i can be defined in which the dependency on the time parameter is made implicit by the following equation.

$$L_1^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i) = \frac{1}{2}[(\dot{\mathbf{Y}}^i)^T \mathbf{1}] \cdot \begin{bmatrix} \Psi^i & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\mathbf{Y}}^i \\ 1 \end{bmatrix}. \quad (40)$$

Finally, from (39) we can obtain a Lagrangian that is *homogeneous* in the rate variables. This is particularly important because the convergence in the algorithm called Hybrid Dynamic Programming (HDP) that our agents use to solve the optimization problem in (32) is accelerated for an homogeneous Lagrangian. We will discuss HDP and this issue in the next section.

We conclude this section by producing yet another equivalent Lagrangian L_2^i which a homogenized version of $L_1^i(\mathbf{Y}^i, \dot{\mathbf{Y}}^i)$. Let $f(\sigma)$ be a positive twice-differentiable function. Then the Lagrangian

$$L_2^i([f, \mathbf{Y}^i], [\dot{f}, \dot{\mathbf{Y}}^i]) = \sqrt{L_1^i(\mathbf{Y}^i, \frac{\dot{\mathbf{Y}}^i}{\dot{f}})} \cdot \dot{f} \quad (41)$$

is positive homogeneous of degree one in the rate. That is, for any positive real number λ ,

$$L_2^i([f, \mathbf{Y}^i], [\lambda \dot{f}, \lambda \dot{\mathbf{Y}}^i]) = \lambda L_2^i([f, \mathbf{Y}^i], [\dot{f}, \dot{\mathbf{Y}}^i]).$$

As we stated earlier, working with homogeneous of degree 1 Lagrangians greatly simplifies the HDP algorithm for computing an optimal solution of problem (42) given below.

$$\min_{\mathbf{Y}^i} \int_0^T L_2^i([f(\xi), (\mathbf{Y}^i(\xi))], [\dot{f}(\xi), \dot{\mathbf{Y}}^i(\xi)]) \cdot d\xi + Y_{2n}^i(T) \quad (42)$$

subject to our initial conditions and positivity constraints. The optimization problem in (42) is our desired continualized formulation!

We want to conclude this section with a prescription of how to obtain an arbitrarily close approximation to the minimal stocking plan from the locally unique solution of (42). This is summarized below as *output procedure*. This is the procedure our MAHCA agents use to generate at each interval the ‘output values’ as a function of the agents internal continualized variables. Although the multi-echelon planner of this paper need not be implemented in real time, our construction would allow such implementation.

3.1 Output Procedure

Let $\mathbf{Y}_*^i(\sigma)$ for $\sigma \in [0, T]$ be the solution to optimization problem (42). Then, we construct the solution to the original problem for each DC*i* and at each discrete time t for $t = 0, 1, \dots, T$ as follows.

1. First get the discrete time coefficient functions $\Theta^i(t)$ and the state estimate $\hat{\mathbf{X}}^i(t)$. Then from (27) find

$$\mathbf{Y}_*^i(t) = \begin{bmatrix} \Theta_*^i(t) \\ \hat{\mathbf{X}}_*^i(t) \end{bmatrix} \text{ for } t \in \{0, 1, \dots, T\}. \quad (43)$$

2. Next using (17), compute feedback expressions for the decision variables \tilde{w}_{i*} and \tilde{u}_{i*} .
3. Finally, using the results of step 2 in equation (18), we can compute the discrete state as a function of the decision variables, the downstream requests and the external demand. We may use the difference between the state estimate constructed from step 1 and the one constructed here, to generate an estimate of the error that may be used for improvement in the computed stock plan.

Figure 4 is a flow chart that summarizes our continualization procedure. As can be noted from the flow chart, this procedure is highly symbolic and can be mechanized for most supply chain type applications.

4 Hybrid Dynamic Programming (HDP)

In this section we give an overview of our HDP algorithm for solving problems of the form of problem (42) [5]. The basis of HDP is an application of Bellman’s principle of optimality to problem (42). First we must construct the following homogeneous function.

$$\beta^i([f(\sigma), \mathbf{Y}_*^i(\sigma)], \mu^i(\sigma)) = \begin{bmatrix} \mu^i(\sigma) \\ L_2^i([f(\sigma), \mathbf{Y}_*^i(\sigma)], \mu^i(\sigma)) \end{bmatrix} \quad (44)$$

where

$$\mu^i(\sigma) = \begin{bmatrix} \dot{\mathbf{Y}}_*^i(\sigma) \\ \dot{f}(\sigma) \end{bmatrix}.$$

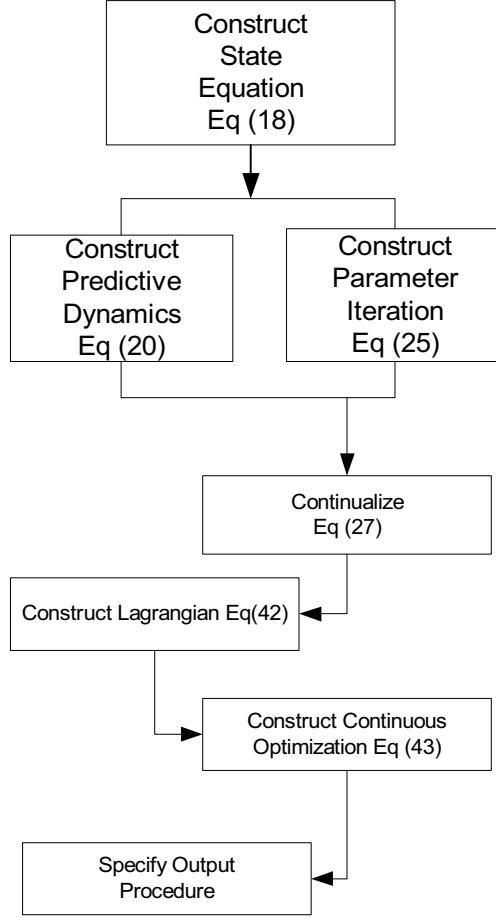


Figure 4: Continualization

Let $V^i(\mathbf{Y}_*^i(\sigma), \sigma)$ be the function defined by

$$V^i(\mathbf{Y}_*^i(\sigma), \sigma) = \min_{\mu^i(\sigma)} \int_{\sigma}^T L_2^i([f(\xi), \mathbf{Y}^i(\xi)], [\dot{f}(\xi), \dot{\mathbf{Y}}^i(\xi)]) \cdot d\xi + Y_{2n}^i(T). \quad (45)$$

Then by Bellman's Principle of optimality, $V^i(\mathbf{Y}_*^i(\sigma), \sigma)$ satisfies

$$-\frac{\partial V^i(y^i, \sigma)}{\partial \sigma} = \min_{\mu^i} \left\{ \frac{\partial V^i(y^i, \sigma)}{\partial y^i} \cdot \beta^i(y^i, \mu^i(\sigma)) \right\} \quad (46)$$

where

$$y^i(\sigma) = \begin{bmatrix} f \\ \mathbf{Y}_*^i \end{bmatrix}.$$

Then by the homogeneity of L_2^i , equation (46) can be written as the following quasi-linear form.

$$-\frac{\partial V^i(y^i, \sigma)}{\partial \sigma} = \min_{\mu^i} \left\{ \frac{\partial V^i(y^i, \sigma)}{\partial y^i} \cdot \frac{\partial \beta^i(y^i, \mu^i(\sigma))}{\partial \mu^i(\sigma)} \cdot \mu^i(\sigma) \right\}. \quad (47)$$

This is the HDP equation for our problem.

From (42), the boundary condition for (47) is given by

$$V^i([T, y^i(T)], T) = \mathbf{Y}_{*2n}^i(T).$$

The solution trajectory for (47) is a geodesic trajectory with respect to the metric defined by $L_2^i([f, \mathbf{Y}^i], [\dot{f}, \dot{\mathbf{Y}}^i])$. This fact has great practical importance for two reasons.

1. We do not have to solve (47) numerically, which is a difficult task.
2. As we shall see, the geodesic is generated via an equation whose functional coefficients encode what we know about the problem. In a real time implementation, the agents would 'learn' over time by tuning these functional coefficients as a function of sensory data. For example, if each agent would actually sample demand of its distribution center, we could use this information to improve the quality of its generated stocking policy.

The trajectory $y_*^i(\sigma) = [f_*(\sigma), \mathbf{Y}_*^i(\sigma)]$ of the optimal solution of (47) is generated by a vector differential equation satisfying the four conditions presented in (48).

For all $j, k \in \{1, \dots, n_i\}$,

(i) $\dot{y}_{*k}^i(\sigma) \neq 0$.

(ii)
$$\frac{\ddot{y}_{*k}^i + 2\Lambda_k^i(y_*^i(\sigma), \dot{y}_*^i(\sigma))}{\dot{y}_{*k}^i(\sigma)} = \frac{\ddot{y}_{*j}^i + 2\Lambda_j^i(y_*^i(\sigma), \dot{y}_*^i(\sigma))}{\dot{y}_{*j}^i(\sigma)} = h(\sigma).$$

(iii) For all $k \in \{1, \dots, n_i\}$, the functions Λ_k^i are positive homogeneous of degree 1 in y_*^i .

(iv)

$$h(\sigma) = L_2(y_*(\sigma), \dot{y}_*(\sigma)). \quad (48)$$

These relations are consequences of the principle of optimality embodied in equation (47) and the homogeneity properties of our constructed Lagrangian. If we define a new evolution variable s by

$$\frac{ds}{d\sigma} = L_2(y_*(\sigma), \dot{y}_*(\sigma)),$$

then we can apply the transformation $\sigma \rightarrow s$ in (48) to obtain the following geodesic equations.

$$\frac{d^2 y_{*k}^i(s)}{ds^2} + 2\Lambda_k^i(y_*^i(s), \frac{dy_*^i(s)}{ds}) = 0 \quad (49)$$

$$i = 1, \dots, n \quad (50)$$

where for $k \in \{1, \dots, n\}$,

$$\frac{ds}{d\sigma} = e^{\int_0^\sigma h(\xi) d\xi}.$$

In the next few paragraphs, we shall develop an equation for the perturbed version of (49) as an algebraic expression in terms of y_{*k}^i and $\frac{dy_{*k}^i(s)}{ds}$, $k \in \{1, \dots, n_i\}$.

For $k \in \{1, \dots, n\}$, we let

$$z_{*k}^i(s) = y_{*k}^i(s) + \epsilon \rho_k^i(s) \quad (51)$$

where $z_{*k}^i(s)$ is a neighboring geodesic of $y_{*k}^i(s)$ in a tubular neighborhood defined by the arbitrarily small parameter $\epsilon > 0$. The central idea is to find necessary conditions in terms of the vector function $\rho^i(s) = [\rho_1^i(s), \dots, \rho_{n_i}^i(s)]^T$ for the tubular neighborhood to be geodesic. Plugging (51) in (49) and using some simple algebraic manipulations, we are lead to the following equation.

$$\begin{aligned} & \epsilon \frac{d^2 \rho^i}{(ds)^2} + 2\epsilon \cdot \frac{\partial \Lambda_k^i(y_*^i(s), \frac{dy_*^i(s)}{ds})}{\partial y_*^i} \cdot \rho^i(s) + \\ & 2\epsilon \cdot \frac{\partial \Lambda_k^i(y_*^i(s), \frac{dy_*^i(s)}{ds})}{\partial \frac{dy_*^i}{ds}} \cdot \frac{d\rho^i(s)}{ds} + o(\epsilon^2) = 0 \end{aligned} \quad (52)$$

where $\frac{o(\epsilon^2)}{\epsilon} \rightarrow 0$ as $\epsilon \rightarrow 0$

Taking the limit as $\epsilon \rightarrow 0$ and some further algebraic manipulation, leads to the following equation.

$$\frac{D^2 \rho^i}{D(s)^2} + \mathbf{Z}^i(y_*^i(s), \frac{dy_*^i(s)}{ds}) \cdot \rho^i(s) = 0 \quad (53)$$

where

(a) $\frac{D}{Ds}$ is a differential type operator called the affine derivative defined by

$$\frac{D\rho^i}{Ds} = \frac{d\rho^i}{ds} + \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} \cdot \rho^i, \quad (54)$$

(b) $\frac{D^2 \rho^i}{D(s)^2} = \frac{D}{Ds} \frac{D\rho^i}{Ds}$, and

(c) $\mathbf{Z}^i(y_*^i(s), \frac{dy_*^i(s)}{ds})$ is the matrix defined by

$$\begin{aligned} \mathbf{Z}^i(y_*^i(s), \frac{dy_*^i(s)}{ds}) &= -\frac{d}{ds} \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} + \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial y_*^i} - \\ & \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} \cdot \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} \end{aligned} \quad (55)$$

This given, we can now state the two conditions for the trajectory corresponding to the solution to equation (53) to be a geodesic corresponding of the dynamic programming equation (47). These conditions are

$$\begin{aligned} (i) \frac{D^2 \rho^i}{D(s)^2} &= \mathbf{0} \\ (ii) \mathbf{Z}^i(y_*^i(s), \frac{dy_*^i(s)}{ds}) &= \mathbf{0} \end{aligned} \quad (56)$$

The vector ρ^i in equation (56) is obtained by integrating a linear equation, namely, equation (53). If the solution is constant along $y_{*k}^i(s)$, then this trajectory is geodesic but not optimal. If the solution is the $\mathbf{0}$ trajectory, then y_{*k}^i is an optimal geodesic.

In our implementation, we use an explicit formula for $\frac{D^2 \rho^i}{D(s)^2}$. This formula is given by (57) below. We note that given the optimal trajectory, y_*^i , the equation for $\frac{D^2 \rho^i}{D(s)^2}$ is linear in ρ^i .

$$\frac{D^2 \rho^i}{D(s)^2} = - \left\{ \begin{array}{l} -\frac{d}{ds} \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} + \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial y_*^i} - \\ \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} \cdot \frac{\partial \Lambda_k^i(y_*^i, \frac{dy_*^i}{ds})}{\partial \frac{dy_*^i}{ds}} \end{array} \right\} \cdot \rho^i \quad (57)$$

The two equations in (55) are the cornerstones for our HDP algorithm. Equation (56)-(i) is used as a test of whether or not the agent's adapter is to be activated. Equation (56)-(ii) is a point-wise algebraic equation in terms the geodesic trajectory $y_*^i(s)$ which is optimal for (47) and its derivatives $\frac{dy_{*k}^i}{ds}$. The MAHCA inferencer uses this equation as the basis for generating these trajectories by extracting a relaxed solution via a version of Newton's algorithm. The entries in the matrix $\mathbf{Z}^i(y_*^i(s), \frac{dy_*^i(s)}{ds})$ are polynomials. Since the domain defined by the positivity constraints is convex, convergence of Newton's algorithm is achieved in a number of iterations for each point that is of the order of the largest order of the polynomials in the matrix. This allows for fast convergence and robust schemas based on Newton's iterative algorithm.

For on-line implementations, deviations from geodicity (optimality) are detected by deviations from zero by the affine derivative vector (56)(ii). For moderate deviations the distance to geodicity in each direction is proportional to the value of the affine derivative in the corresponding direction. This is the adaptive mechanism the agents use.

We conclude this section with a block diagram that summarizes the flow of the HDP procedure.

References

- [1] Kohn, W., and Rummel, J.B., "Multi-Echelon Inventory Planning System I: A Multiple Agent Hybrid Control Implementation", Proceedings of World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), vol. XVII, (2001), Orlando, Florida, pp. 227-232.

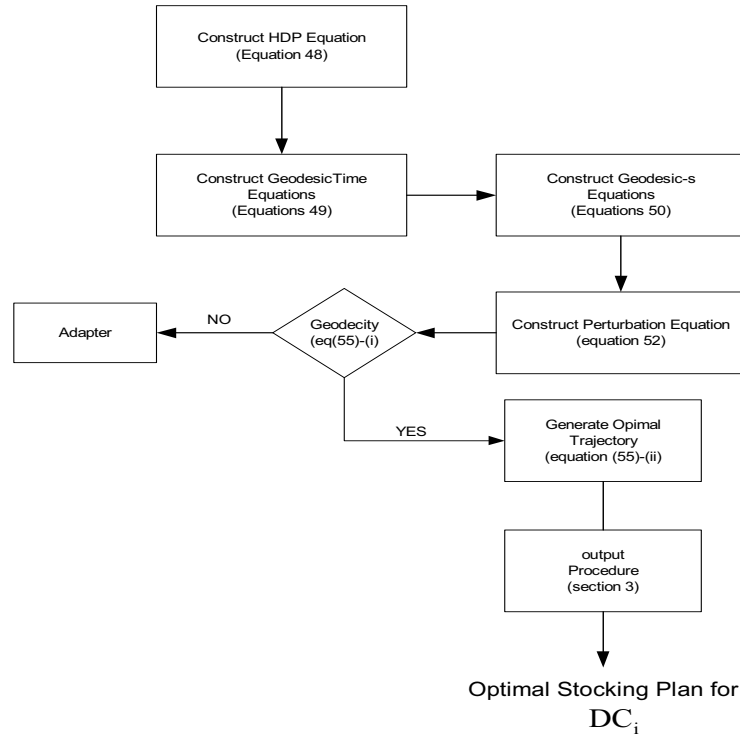


Figure 5: HDP Procedure

- [2] Kohn, W., and Rummel, J.B., “Multi-Echelon Inventory Planning System II: Continualization”, Proceedings of World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), vol. XVII, (2001), Orlando, Florida, pp. 233-238.
- [3] Grossman, R.L., Nerode, A., Ravn, A. and Rischel, H. eds., *Hybrid Systems*, Lecture Notes in Computer Science 736, Springer-Verlag, (1993).
- [4] Antsaklis, P., Kohn, W., Nerode, A, and Sastry, S. eds., *Hybrid Systems II*, Lecture Notes in Computer Science vol. 999, Springer-Verlag, (1995).
- [5] Kohn, W., Nerode, A., and Rummel, J.B., “Feedback Derivations: Near Optimal Controls for Hybrid Systems,” Proceedings of CESA’96 IMACS Multiconference, (1996), 507-511.
- [6] Wolf Kohn, Jeffrey B. Rummel, Anil Nerode, and John James, “Multiple Agent Hybrid Control for Manufacturing Systems,” Proceedings of the 1996 IEEE International Symposium on Intelligent Control (1996), 348-353.
- [7] Wolf Kohn and Jeffrey B. Rummel, “Digital to Hybrid Program Transformations,” Proceedings of 1996 IEEE International Symposium on Intelligent Control, (1996), 342-347.

- [8] Wolf Kohn, Anil Nerode, and Jeffrey B. Rummel, “Continualization: A Hybrid Systems Control Technique for Computing,” Proceedings of CESA’96 IMACS Multiconference, (1996), 517-521.
- [9] Clark, A.J. and Scarf, H, “Optimal Policies for a Multiechelon Inventory Problem,” Management Science vol. 6, 465-490.

A Problem Formulation

Multi-Echelon Inventory Planning

Consider a directed graph $G = (V, A)$ which represents the distribution network for a product p . Each vertex v in the set V represents a distribution center (DC). The existence of an edge (v, v') implies that the DC represented by v acts a source of supply for the DC represented by v' . When a DC can receive supply from multiple sources, it can be assumed that there is a prespecified order in which the sources will be polled for supply. The lead-time between DC’s is one time period. There is one infinite supply source which supplies two nodes in the network. Demands for item p at vertex v in time period t follow a distribution whose c.d.f is given by $F_v^t(\cdot)$. The demand is assumed to be greater than zero in every time period. The cost of holding inventory at v per period is h_v . The penalty for not satisfying demand is s_v per unit of demand. The transportation lead-time from the infinite source external supplier is given by $l'_{vv} (> 1)$ time periods and is deterministic. The objective is to determine the stocking plan that minimizes expected costs.

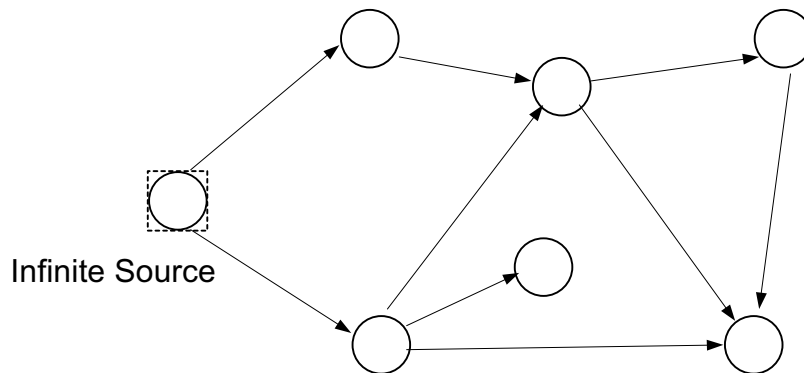


Figure 6: System Graph

B Description of MAHCA

B.1 Agent Framework

The general framework in which MAHCA operates is shown in Figure 7. The framework is composed of three entities: a distributed process under control or planning, a collection of controlling-planning agents and a communication network for agent synchronization. In the system of this paper, the process is the multi-echelon system, an agent is the device implementing the planning policy for each DC i , and the agent network is the network dictated by the network of DC's.

Each agent in the framework interacts with the process via its own sensors and actuators. An agent generates control actions that it computes in real time. The agent's control law generates the control actions. An agent's control law is constructed in real time and reflects the desired behavior of the process in accordance with stored requirements and as a function of observed sensory and inter-agent data flowing through the communication network. The data flowing through the communications network consists of inter-agent constraints used to maintain synchronization and to re-acquire it if events in the process cause its loss.

B.2 Functional Elements of MAHCA

MAHCA is a software system for the autonomous synchronization and control of distributed real-time processes. >From an operational point of view, the model shown in Figure 7 can represent the distributed process under control and the MAHCA system carrying out the control.

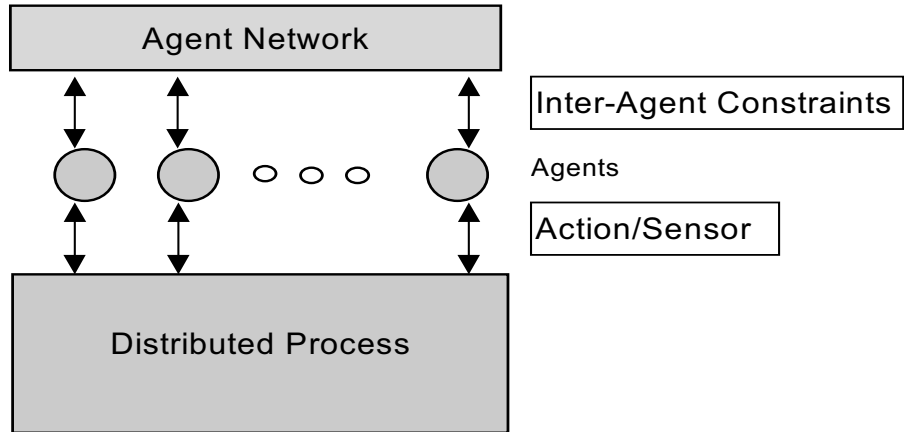


Figure 7: MACHA Framework

Each of the circles represents an agent. An agent is a logic device that carries out prespecified synchronization and/or control functions. The action of each agent is a function of three information items:

- (a) **Sensory Data:** On-line status data flowing from the process to the agent.
- (b) **Active Knowledge:** Selected information data encoded in the agent's Knowledge Base.
- (c) **Inter-agent constraints:** On-line status information from other agents via the prespecified logic network.

An agent carries out its control and synchronization functions by issuing command actions to the process and constraint data to the other agents.

The framework proposed in Figure 7 is very general. It is adequate for the representation of many dynamic distributed processes as well as their control and synchronization activities. For example, consider a discrete multi-component manufacturing process. The process is carried out by an assembly line composed of assembly stations and product transportation subsystems. Each assembly station performs a partial assembly task on the incoming items, which are then directed by the appropriate transportation subsystem to the station responsible for carrying the next stage in the assembly. In this scenario each assembly station and transportation subsystem carries out its tasks under the command or supervision of an assigned agent. The agent knows about the dynamics, constraints and operating rules of its station from encoded knowledge in its knowledge base. It knows about the current status of the station from the sensory information. It acquires and receives synchronization information from the other agents in the form of imposed constraints on the actions it can select.

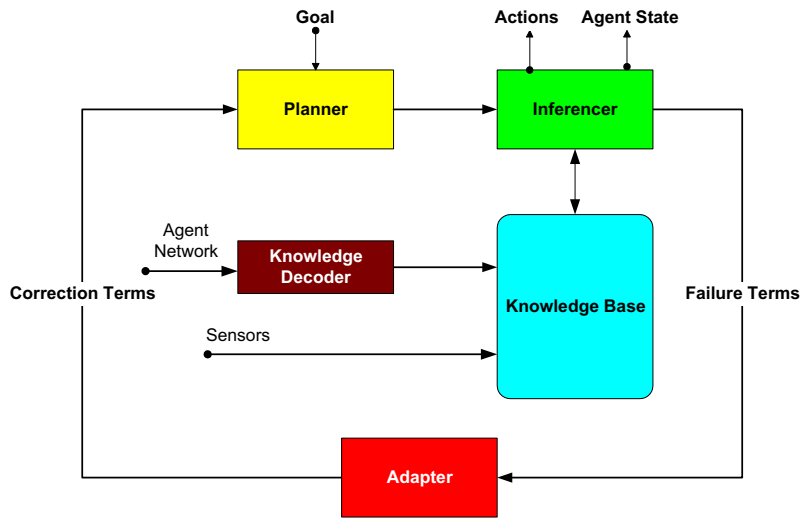


Figure 8: Agent Architecture

An agent's functionality is implemented through an architecture called the Multiple Agent Hybrid Control Architecture (MAHCA). The agent architecture operates

via two interacting asynchronous loops: the *control loop* and the *reactive learning loop*. The control loop generates control actions and the agent's state as a function of its current knowledge to satisfy an internally generated plan. The reactive learning loop modifies the agent's plan as a function of observed agent behavior. In the current prototype, these two loops are implemented via five interacting modules: a *Planner*, an *Inferencer*, a *Knowledge Base*, and *Adapter*, and a *Knowledge Decoder*. The agent architecture is shown in Figure 8. Below we list the functionality of each module in the architecture. *Planner* constructs and repairs the agent's optimization criterion. *Inferencer* determines whether there is a nonempty solution set for the agent's optimization problem. If there is such a solution set, it infers the appropriate control actions, new state information, and inter-agent information. *Knowledge Base* stores and updates the agent's knowledge. *Adapter* repairs failure terms and computes correction terms. *Knowledge Decoder* receives and translates data from the other agents.