A Neuro-Dynamic Programming approach to synthesize optimal dispatching rules in logistics

Mauro Boccadoro ∗ Francesco Martinelli ∗∗

Abstract: A dynamic model of a logistic node of a transportation network is presented and a performance index comprising transportation and inventory costs is considered. The optimization problem is solved considering a class of policies called full capacity. The derivation of optimal dispatching rules by integer programming or dynamic programming is usually intractable in realistic scenarios. For this reason, a neuro-dynamic programming (NDP) approach is pursued. Several simulation results are discussed, in particular for what concerns the role of transportation costs in the characterization of the optimal policies.

Keywords: Logistics, Neuro-dynamic programming, Transportation networks

1. INTRODUCTION

The management of a logistic node in a transportation network is a complex problem where several factors have to be taken into account, from the availability of carriers and their assignment to particular tasks (in terms of products to be shipped, destinations or routes), and fulfillment of various performance criteria such as timely delivery, minimization of transportation and inventory costs; see, among others, Crainic [2002], Carvalho et Powell [2000], Powell et al. [2001], Xu et Hancock [2004]. In the literature, many instances of decisional problems for these systems have been presented and solved; often transportation problems can be addressed in terms of linear programming problems, see, e.g., Powell et Carvalho [1998,b], Yang et al. [2004], Dall’Orto et al. [2006], and developing ad-hoc techniques to obtain the solutions, such as dynamic programming with linear approximation of the (unknown) value function. It is worth mentioning that by the approach of Powell et Carvalho [1998,b] the framework of the Logistic Queuing Networks is introduced. A slightly different paradigm considers shipping policies for simplified models of a logistic network (or a part of it) and addresses the minimization of transportation and inventory costs, see for example Speranza et Ukovich [1994], Bertazzi [2008] and Kang et al. [2008] where a stochastic setting is adopted.

A simplified model of a logistic node in a transportation network was introduced in Boccadoro et al. [2009, 2010] and a necessary and sufficient condition for the stability of the inventory was given. Some feedback stable policies were simulative analyzed in Boccadoro et al. [2009] and, neglecting the transportation cost, a Neuro-Dynamic Programming approach was introduced in Boccadoro et al. [2010] to derive optimal expedition strategies.

The possibility of neglecting the transportation cost in the optimization problem is reasonable under some particular circumstances, including the class of policies considered and the operating conditions of the node (traffic flow, number of vehicles). The investigation of the effect of the transportation cost on the performance index under more general conditions is one of the objective of the present paper. Optimal dispatching policies can be derived either by solving an integer programming problem or, as in this paper, by considering a dynamic programming approach. Value iteration or policy iteration techniques, however, fail to solve the problem in realistic scenarios, for the well known Bellman’s curse of dimensionality. For this reason, an approximate approach is considered to tackle the problem, based on Neuro-Dynamic Programming (Bertsekas et Tsitsiklis [1996]): this is a well established framework, that addresses the complex task of deriving the value function by using learning techniques, like, e.g., Neural Networks. We propose as approximation architecture for the value function a linear combination of the state variables, choice motivated by theoretical considerations and validated through numerical computations. Numerical results will be presented to illustrate the effectiveness of the proposed approach.

2. PROBLEM FORMULATION

Consider a logistic node collecting items to be shipped to P different locations, and let \( x_i(k) \geq 0, k = 0, 1, \ldots \), be the quantity of items with destination \( i = 1, \ldots, P \), stocked at the node at time \( t_k \). A round-trip time \( T_i \) is associated to each destination \( i \). The evolution of each \( x_i \) is observed at various decisional times \( t_k \), and as such characterized by a discrete time dynamics. Denoting \( d_i(k) \) the amount of goods to be sent to destination \( i \) arriving in the node in the interval \( [t_k, t_{k+1}) \) and \( u_i(k) \) the amount of goods shipped to destination \( i \) in the same interval, we have:

\[
x_i(k + 1) = x_i(k) + d_i(k) - u_i(k) \tag{1}
\]
For simplicity of notation we assume $t_k = k$ for all $k$ with $T_i \in \mathbb{N}$. The dynamics of the vehicles executing the shipping task is given in terms of $n_i(k)$, defined as the number of vehicles that during the interval $(k, k+1)$ are at a time distance $\ell$ from the base (so that $\ell \in \{0, 1, \ldots, T_{\text{max}}\}$, with $T_{\text{max}} = \max_i (T_i)$, and $\ell = 0$ denotes presence at the base), in terms of $\nu_i(k)$, the number of vehicles assigned to destination $i$, and of $r(k)$, the number of vehicles arriving from outside, in the interval $(k, k+1)$. Accordingly, the total number of vehicles available for a shipping task at time $k$ is given by $n_i(k) := n_0(k) + r(k)$. Considering the nominal scenario where a fixed quantity $N_c$ of vehicles serves the logistic node, we have $r(k) = \sum_{i=1}^P \nu_i(k - T_i)$, i.e. the number of vehicles coming back from their expedition. Defining the vector $n := [n_0, \ldots, n_{T_{\text{max}}}]^T$ and $\nu = [\nu_1, \ldots, \nu_P]^T$, the dynamics may be expressed as:

$$n(k+1) = An(k) + B\nu(k)$$

where $A$, except for a 1 in position $(1,1)$, is a delay matrix (i.e., the Jordan block for the null eigenvalue of order $T_{\text{max}} + 1$), whereas $B$ is composed by all null elements except for $B_{1,j} = -1, \forall j$ and for $B_{T+1,i} = 1$. Several extensions of this nominal model could be considered, see Boccadoro et al. [2010] for more details.

In synthesis, the state of our system is given by $z = [z^T, n^T]^T$, whereas the decision variables are $w = [w^T, u^T]^T$, with $x$ and $u$ (column) vectors comprising all the $x_i$, and $u_i$, respectively. Denoting by $Z$ the state space, notice that the configurations described by $n$ are constrained by $\sum_{i=1}^P n_i = N_c$. Let shortly define by $f$ the function in (1) and (2) which gives the successor $z(k+1)$ of $z(k)$ under control $w(k)$, i.e., $z(k+1) = f(z(k), w(k))$.

The optimal control of this system is defined in terms of the following performance index:

$$J = \sum_{k=0}^\infty g[z(k), w(k)]\gamma^k$$

where $\gamma \in (0,1)$ is a discount factor. The function $g(.)$ may penalize inventory in the node (as a measure of timely expeditions of the relative items) and also include transportation costs, for example:

$$g(z, w) = \sum_{i=1}^P (h_i \nu_i + c_i x_i)$$

for proper coefficients $c_i$ and $h_i$. The first term in the cost function implicitly accounts also for the optimal composition of the load for the various vehicles. As a first attempt to tackle this problem, we will consider it under the hypothesis that all vehicles travel always fully loaded. This assumption is reasonable under large transportation costs and/or heavy traffic conditions (where allowing the possibility of sending partially loaded vehicles may even compromise the stability), but may become significantly sub-optimal in the case of reduced inflow rates, large holding costs $c_i$ and small travel costs, where it could be convenient to send partially loaded vehicles in order to reduce the inventory stock.

The possibility of completely loading a vehicle is justified by the fluid approximation of the stock variables $x_i$ considered here, assuming the unit volume represented by the (equal) capacity of each vehicle. The quantities $n$ and $\nu$ will thus represent multiples of the unit volume.

A further simplification will consist, as mentioned, in restricting the problem to the following class of policies, which may not include in general the optimal control.

**Definition 1.** A full capacity policy is a feedback rule which allows only expeditions with full vehicles and never leaves a vehicle idle if for at least one destination there is enough material to fill a vehicle. □

Under the full load assumption and the fluid approximation, for each destination $i = 1, \ldots, P$ and for all $k$, the amount of goods $u_i(k)$ shipped to destination $i$ at time $k$ coincides with the number of vehicles $\nu_i(k)$ sent to that destination, i.e. $u_i(k) = \nu_i(k)$.

We report a stability result derived in Boccadoro et al. [2009], which also provides the minimum number $N_{\text{stab}}$ of vehicles to meet the transportation demand.

**Theorem 1.** Under the assumption of constant demand process (i.e. $d_i(k) = d_i$ for all $k = 0, 1, \ldots$), the condition

$$N_{\text{stab}} := \sum_{i=1}^P d_i T_i \leq N_c$$

is necessary and sufficient to stabilize the node (i.e. to maintain all buffers bounded). In addition, under (5), any full capacity policy would attain the stability. □

The stability condition, together with the assumption that only full capacity policies are considered, guarantees time delivery regardless of cost parameters. In fact, if not restricting the problem to full capacity policies, under large transportation costs, the system may start to send vehicles only when the buffers reach a large threshold. More details on this will be reported in Section 5.

### 3. THE ROLE OF THE TRANSPORTATION COST

In Boccadoro et al. [2010], the transportation cost was neglected: this was motivated by considering that if the transportation cost is paid on a day per day basis and only full capacity policies are considered, in many cases it represents a fixed component, i.e. it is independent of the particular full capacity policy considered.

In this section we deal with the role of transportation costs by considering two different modes of paying the traveling expenses, both modeled by (4) for proper coefficients $h_i$. The first case consists in paying all the transportation cost at the beginning of the travel. This corresponds to take $h_i = h \cdot T_i$ in (4), where $h > 0$ is a constant modeling, e.g., the traveling per unit time expenses. In this case, the transportation cost component in (3) can be written as:

$$J_{11} = h \sum_{k=0}^\infty \sum_{i=1}^P \nu_i(k) T_i \gamma^k$$

Due to the discount factor $\gamma^k$, it seems preferable to push toward the future expeditions associated with far destinations (this will be analyzed afterwards). If the traveling cost is paid on a *day per day basis*, it should be taken $h_i = h \sum_{n=0}^{T_i-1} \gamma^n = h \frac{1 - \gamma^{T_i}}{1 - \gamma}$ with the transportation cost component in (3) given by:

$$J_{12} = h \sum_{k=0}^\infty \sum_{i=1}^P \nu_i(k) \frac{1 - \gamma^{T_i}}{1 - \gamma} \gamma^k.$$  

(6)
i.e., paying on a day by day basis, the first day traveled costs $h$, the second $h \cdot \gamma$, and so on. The following lemma, which can be proved by observing that the number of traveling vehicles in each day $k$ is given by $N_c - n_0(k)$, provides another expression for the cost $J_{t_2}$.

**Lemma 1.** If the traveling cost is paid on a day by day basis, the transportation cost in (3) is given by:

$$J_{t_2} = h \sum_{k=0}^{\infty} (N_c - n_0(k)) \gamma^k \quad \square$$

Using this expression it is easy to verify the following facts:

1) If $N_c \gg N_{stab}$ (in the sense that items never wait for vehicles), $J_{t_2}$ and $J_{t_1}$ take the same value under any full capacity policy. This is because the expeditions are conditioned only on the buffer values: all the buffers which fill at least a vehicle are sent out of the node, hence any full capacity policy will be characterized by the same expeditions (and also by the same total cost $J$).

2) If $N_c < N_{stab}$, all vehicles are always traveling for all $k$. Hence $J_{t_2} = h \cdot N_c / \gamma$ will be independent of the particular full capacity policy adopted (being $n_0(k) = 0$ for all $k > 0$ and using the expression of $J_{t_2}$ in Lemma 1), while $J_{t_1}$ can be policy dependent, since long expeditions executed at the beginning imply larger total transportation costs with respect to the case they are postponed to the future.

3) If $N_c \approx N_{stab}$ (but $N_c > N_{stab}$), both $J_{t_1}$ and $J_{t_2}$ are policy dependent. If starting from large buffer contents, at the beginning the situation is similar to the one described in item 2 above (lack of vehicles, i.e. $n_0(k) = 0$ for a while). At steady state, on the contrary, the situation may become similar to the one reported in item 1 above. A difference in $J_{t_1}$ and, especially, in $J_{t_2}$ is mainly caused in this case by the steps around the time the steady state is reached, as shown subsequently.

**Example 1.** Consider a system with $P = 2$ destinations, demand $d_1 = d_2 = 1$, transportation times $T_1 = 1$ and $T_2 = 10$, $h = 1$ and initial condition $x_0 = (12, 12)$. Here $N_c = 12$, while $N_{stab} = 11$. We have considered two different full capacity policies on a simulation interval of 50 steps. The expeditions of the two policies have been generated randomly: the first one starts by shipping items of the first buffer (i.e. assigning all the 12 vehicles to destination 1), in the second case the items of buffer 2 are processed first. We obtain in both cases $n_0(k) = 0$ for all $k$ (this corresponds to item 3 above with large initial conditions: the steady state has not yet been reached at $k = 50$), with $J_{t_2}$ coincident in fact under the two policies. As for the cost $J_{t_1}$, we have $J_{t_1} = 145.9$ in the first case and $J_{t_1} = 150.6$ in the second, showing how it is better to postpone expeditions to long destinations (if the transportation cost is concerned). Here and in all the numerical examples considered in this paper $\gamma = 0.9$.

**Example 2.** Consider now a system with $P = 2$, $d_1 = d_2 = 1$, $T_1 = 1$ and $T_2 = 3$, $h = 1$ and starting from the initial condition $x_0 = (2, 4)$. Here $N_{stab} = 4$ and we have considered $N_c = 5$. In Fig. 1 it is reported the evolution of $n_0(k)$ under two different full capacity policies. It is possible to verify that, even if the vehicles remaining idle in the node are different under the two policies, the total number of expeditions is the same (since the buffer contents at the end of the simulation are the same under the two policies), in such a way that if an average transportation cost (rather than a discounted one) was considered, the two policies would provide the same traveling expenses. The transportation costs are given by $J_{t_1} = 39.84$ and $J_{t_2} = 36.53$ for the first policy and by $J_{t_1} = 40.04$ and $J_{t_2} = 36.69$ for the second. This example falls in the situation described in item 3 above, where the steady state is reached after a few steps.

**Example 3.** Consider the system of Example 2, but with a larger initial state $x_0 = (10, 10)$. The evolution of $n_0(k)$ under two different full capacity policies is illustrated in Fig. 2. In this case it is more evident what mentioned in item 3 above, that at the beginning, due to the large backlog, the behavior is similar to the case of a lack of vehicles while at the end the situation corresponds to an over dimensional situation, with a vehicle permanently unused. The difference in the number of vehicles used by the two policies appears around time step 34, i.e. when the system is reaching its steady state. The costs are given by: $J_{t_1} = 43.75$ and $J_{t_2} = 40.19$ for the first policy and by $J_{t_1} = 43.58$ and $J_{t_2} = 40.18$ for the second. The almost coincidence of $J_{t_2}$ (and actually also of $J_{t_1}$) depends on the fact that the difference in $n_0(k)$ occurs only at steps 33–34, with a very small $\gamma^3$ discount factor.

**4. SOLUTION APPROACH**

The task to derive the optimal policy for the system presented requires approaches able to tackle the state space combinatorial complexity. In fact, if we discretize the buffer content $x_i$ in order to apply a dynamic programming algorithm, and if $m_x$ denotes the number of discretized values for each buffer (i.e. $x_i \in \{0, 1, \ldots, m_x\}$) the number of discrete states is $|Z| = (m_x + 1)^P \cdot C(N_c + T_{max}, N_c)$, where $C(n, k)$ is the binomial coefficient. For example, if $P = 10$, $N_c = 20$, $T_{max} = 3$ and $m_x = 50$, $|Z| = 2.1 \cdot 10^{20}$.

An approach based on Neuro-Dynamic Programming (NDP) can be pursued, where a general structure is assumed for the value function, which is recursively approximated during the execution of the algorithm, via

![Fig. 1. Unused vehicles in Example 2](image1)

![Fig. 2. Unused vehicles in Example 3](image2)
reinforcement learning (Bertsekas et Tsitsiklis [1996]). We apply this idea considering the Policy Iteration paradigm (which usually converges to the optimal policy in very few iterations, unlike a value iteration approach). The Policy Iteration method consists of the repetition of two steps, starting from an initial tentative feedback policy $\pi$: (i) evaluation of the value function $V^\pi$ associated with policy $\pi$ (this can be done analytically or via simulation in the discounted scenario of this paper), (ii) policy improvement.

More in detail, if $f_\pi(z) := f(z, \pi(z))$ is the state reached under policy $\pi$ from $z$ and $f^k_\pi(z)$ denotes the state reached from $z$ after $k$ steps under that policy (with $f^0_\pi(z) = z$), the computation of the value function $V_\pi$ associated with a certain policy $\pi$ can be performed via simulation by:

$$V_\pi(z) = \sum_{k=0}^{K} \gamma^k g(f^k_\pi(z), \pi(f^k_\pi(z)))$$  \hspace{1cm} (7)

where $K$ is a certain (discrete) time horizon, such that the sum of the terms for $k > K$ can be neglected (this is possible in view of the discount factor $\gamma$). Subsequently, the policies are iteratively improved according to the following greedy mechanism:

$$\pi_{i+1}(z) = \arg\min_w \{g(z, w) + V_{\pi_i}[f(z, w)]\}$$  \hspace{1cm} (8)

The classical version of this algorithm needs that both (7) and (8) must be computed for all $z \in Z$; however, to make the problem tractable, the NDP based techniques devise the approximation of the value function based on the exact knowledge of only some samples of $V$, i.e., (7) is evaluated only in a (small) subset of $Z$, and based on these data and on an approximation scheme, an estimate $\hat{V}$ of $V$ is computed for all $z \in \mathbb{Z}$. Accordingly, the algorithm (7)-(8) is modified as follows:

Algorithm 1. Approximate Policy Iteration.

Set $i = 0$ and select a tentative policy $\pi_0$ (e.g. a Clear the Largest Buffer policy).

Step 1) Compute via simulation as in (7) (with $\pi = \pi_i$) the Value Function $V_{\pi_i}(z)$ associated to $\pi_i$ for all $z \in \mathbb{Z}$, where $\mathbb{Z}$ is a small set of states, i.e. $|\mathbb{Z}| \ll |\mathbb{Z}|$. At step 0, $f_{\pi_0}(z)$ is known for all $z$. At step $i > 0$, $f_{\pi_i}(z) = f(z, \arg\min_w \{g(z, w) + \hat{V}_{\pi_{i-1}}[f(z, w)]\})$.

Step 2) Given $V_{\pi_i}(z)$ for $z \in \mathbb{Z}$, determine through a certain functional architecture (e.g. a neural network), an approximation $\hat{V}_{\pi_i}(z)$ of $V_{\pi_i}(z)$ for all $z \in \mathbb{Z}$.

Step 3) If $\pi_i = \pi_{i-1}$ (this can be easily checked by comparing the parameters of the last two iterations in the approximation architecture of the value function), the NDP policy $\hat{\pi}(z)$ is implicitly given by the last obtained approximate value function $\hat{V}$ through the equation $\hat{\pi}(z) = \arg\min_w \{g(z, w) + \hat{V}[f(z, w)]\}$. Otherwise set $i = i + 1$ and go to Step 1.

The considered approach is no more guaranteed to converge to the optimal policy, due to the approximation introduced in Step 2. Since the cost function (4) is linear and the cost index (3) is discounted, it is reasonable to approximate the value function through a linear combination of the state variables, i.e. $\hat{V}_{\pi_i}(z) = p_i z$, where $p_i \in \mathbb{R}^{P + T_{\text{max}} + 1}$ is a (row) vector of parameters. The functional approximation is achieved by a Newton method algorithm which identifies the $p_i$ which fits best the $V_{\pi_i}(z)$ to the $V_\pi$ sampled in $\mathbb{Z}$, in the sense of the minimization of

$$\sum_{z \in \mathbb{Z}} (V_{\pi_i}(z) - p_i z)^2$$

Since a vehicle sent at time $k$ to a destination with $T_i = 1$ will be already available at time $k + 1$ for a new expedition, the state variables $n_0$ and $n_1$ can be lumped in a unique variable $n_0 + n_1$ to compute the value function. For this reason, in the following, $p_i$ will be considered in $\mathbb{R}^{P + T_{\text{max}}}$, and it will multiply a modified vector $z$ where the element $P + 1$ is $n_0 + n_1$.

In Boccadoro et al. [2010] the performance of the solution found through the NDP approach was compared with the $c\mu$ rule, a myopic policy inspired by the manufacturing system literature based on a priority among the buffers. This myopic policy (which was proved to be optimal in Martinelli et al. [2001] for flexible production systems) adapted to the present context consists in processing the buffers according to a priority established by the product of the cost coefficient $c_i$ in (4) and the maximum processing capacity for items of type $i$, which is given by $\mu_i = N / T_i$.

In Boccadoro et al. [2010], where the transportation cost was not considered, it was shown that this heuristic is not optimal. As the transportation component is not handled in the $c\mu$-rule, we expect that the performances of the heuristic become worse when traveling costs are included in the performance index (as illustrated in Section 5).

5. NUMERICAL RESULTS

The NDP approach is applied to a numerical instance of the problem. As starting policy $\pi_0$ in the NDP algorithm 1, a (full capacity) Clear the Largest Buffer (CLB) policy is chosen; according to this policy vehicles are assigned at each step to the buffer having the largest content.

The considered instance is characterized by $m_2 = 20$, $P = 2$, $N_c = 5$, $c_1 = 5$, $c_2 = 16$, $T_1 = 1$ and $T_2 = 3$ (so that $|\mathbb{Z}| = 24696$). A penalty has been added when a buffer exceeds its maximum content. This problem is still tractable and has been solved by ordinary value iteration so that the optimal value function $V^*(z)$ and the optimal decisions $\pi^*(z)$ are available for all $z \in \mathbb{Z}$ for comparison purposes. Running Algorithm 1 with a randomly selected set $\mathbb{Z}$ of 33 elements, convergence is achieved in few steps.

Let $\hat{V} = \hat{p} \cdot z$ denote the parameterized value function obtained after convergence (i.e. $p_i \rightarrow \hat{p}$) and let $V_{\pi}$ be the actual cost associated with the obtained policy $\pi$ (computed through (7) with $K = 100$, where $\gamma^K \approx 10^{-5}$). This policy $\pi$ will be referred to in the following for brevity as the NDP policy.

In Fig. 3 the behavior of the two heuristics mentioned in the paper, the CLB policy on the left and the $c\mu$ rule on the right, is illustrated as follows: assuming all the vehicles available at the base (i.e. $n_0 = N_c = 5$), the plots report for all buffer contents $(x_1, x_2)$ the number of vehicles assigned to destination 2 and a white box all vehicles assigned to destination 1 while a gray box means that some vehicles are assigned to destination 1 and the others to destination 2.
2: actually the figure shows the quantity $\nu_1 - \nu_2$, which ranges from $-5$ (black) to 5 (white). The CLB assigns as expected all the vehicles to the destination with larger backlog (left plot of Fig. 3) while the $c_\mu$ rule (right plot of Fig. 3) selects the destination with higher index $c_\mu$, i.e. destination 2, being $c_2/T_2 = 5.3$ and $c_1/T_1 = 5$ (clearly, when $x_2 < 5$, some vehicles must be assigned to destination 1).

Fig. 3. CLB (left) and $c_\mu$ (right) choices when all vehicles are in the node (i.e. for $n = (5,0,0,0)$)

Fig. 4. Optimal (left) and NDP (right) choices for $h = 1$ (up) and $h = 100$ (down) with $n = (5,0,0,0)$

Now, assume that in the cost function (4) $h_i = hT_i$ with $h = 1$ (this corresponds to the $J_{clb}$ index described in Section 3). This transportation cost turns out to be negligible with respect to the backlog cost (actually we have obtained the same results of the case $h = 0$) and in fact, the optimal policy $\pi^*$ (computed through an exhaustive value iteration) is very similar to the $c_\mu$ rule, as found for the $h = 0$ case in Boccadoro et al. [2010]. This can be observed by comparing the right plot of Fig. 3 with the left-up plot of Fig. 4 (generated according to the same notation described for Fig. 3).

The histograms in Fig. 5 have been generated as follows: plot (a) refers to the NDP policy. If $V_\pi$ denotes the cost of the NDP policy, plot (a) reports the number of states $z$ giving the deviation w.r.t. the optimal cost $\Delta V_\pi := V_\pi/V_\pi^*\%$ (reported on the $x$-axis). Similarly, in plots (b) and (c), the histogram is reported with respect to the deviations $\Delta V_{clb} := V_{clb}/V_\pi^*\%$ and, respectively, $\Delta V_{c_\mu} := V_{c_\mu}/V_\pi^*\%$, being $V_{clb}$ and $V_{c_\mu}$ the cost of the CLB and of the $c_\mu$ policies. Plot (d) compares the $c_\mu$ rule with the NDP policy by reporting the histogram of $\Delta V_{c_\mu-\pi} := V_{c_\mu-\pi}/V_\pi^*\%$. Hence, from plot (c) of Fig. 5, it can be observed, as mentioned, how the optimal policy in this case of $h = 1$ looks similar to the $c_\mu$ rule (the optimal $V_\pi$ and the $c_\mu$ cost $V_{c_\mu}$ are almost coincident, with almost all the states associated with a 0 deviation of $V_{c_\mu}$ w.r.t. $V_\pi$). Moreover, from the plot (a) of this figure, it is apparent how also the NDP algorithm (initialized with the CLB policy) produces an almost optimal policy (which in fact, from plot (d), is almost coincident with the $c_\mu$ rule), while the CLB is characterized by several states associated to a non optimal cost (see plot (b) of Fig. 5). The similarity of the NDP policy with the optimal policy is also evident in Fig. 4, where the choices of the optimal policy (up-left) are like those performed by the NDP policy (up-right).

Consider now a large transportation cost, by taking $h_i = hT_i$ with $h = 100$ in the cost function (4). This large transportation cost makes destination 1 more appealing than the other destination (characterized by a larger transportation time) in such a way that the $c_\mu$ rule (which gives priority to destination 2) becomes less satisfactory than in the $h = 1$ case. This can be observed in the down-left plot of Fig. 4 which reports the transportation choices of the optimal policy $\pi^*$: it is evident how this policy differs from the $c_\mu$ rule, illustrated in the right plot of Fig. 3. In other words, for this numerical setting, the transportation cost makes destination 1 the most appealing.
where the deviation of the $c\mu$ rule from the optimum (plot (c)) is more evident than the deviation from the optimum of the NDP policy (plot (a), see also plot (d) for a direct comparison of the two policies). The CLB choices in this case are more effective than in the $h = 1$ case (plot (b)).

**Fig. 6.** Histograms for the case $h = 100$

It should be mentioned that $h = 100$ is a large transportation cost, in the sense that if the constraint on full capacity policies was removed, the optimal policy would never send vehicles (at least in our numerical example where buffers are bounded, otherwise it would start to send vehicles after the buffers have reached some large threshold). On the contrary, the full capacity assumption implies that buffer levels remain small even if the transportation cost rises, guaranteeing time delivery.

For all values of $h$ for which it remains convenient to send vehicles, the performance of the $c\mu$ rule is not far from that of the optimal policy and the NDP algorithm does not provide significantly better results: this is due to the fact that when the transportation costs are quite small with respect to the inventory costs, the $c\mu$ rule still represents a good heuristic, which could be improved through NDP only by more skillful approximation schemes. Notice that, at this regard, using the $c\mu$ rule as initial guess $\pi_0$ in Algorithm 1 would not provide a sensible improvement in the final policy $\hat{\pi}$.

6. CONCLUSIONS

In this paper, a simplified model of a logistic node has been considered, where items arrive from outside to the node and must be routed to different destinations. To optimally control this system, a neuro dynamic programming (NDP) approach is used in view of the combinatorial complexity of the problem, which cannot be faced through an exhaustive dynamic programming computation. This optimization problem has been studied by considering full capacity policies, i.e. feedback rules that only allow expeditions with full vehicles and never leave a vehicle idle if there is enough material to fill it. The results of this work extend those previously reported in Boccadoro et al. [2010] by investigating the role of transportation costs. In particular, the results obtained validated the adoption of a sub-optimal feedback control inspired by the manufacturing literature when transportation costs are negligible, whereas when this contribution to the overall cost rises, this heuristic may become significantly suboptimal.

REFERENCES


Boccadoro, M. and F. Martinelli. “Control of a logistic node via Neuro-Dynamic programming” *49th conference on Decision and Control* (CDC 2010), Atlanta, Georgia (USA), December 2010.


