

Hierarchical routing control in discrete manufacturing plants via model predictive path allocation and greedy path following

Lorenzo Fagiano, Marko Tanaskovic, Lenin Cucas Mallitasig, Andrea Cataldo and Riccardo Scattolini

Abstract— The problem of real-time control and optimization of components’ routing in discrete manufacturing plants is considered. This problem features a large number of discrete control inputs and the presence of temporal-logic constraints. A new approach is proposed, with a shift of perspective with respect to previous contributions, from a Eulerian system model that tracks the state of plant nodes, to a Lagrangian model that tracks the state of each part being processed. The approach features a hierarchical structure. At a higher level, a predictive receding horizon strategy allocates a path across the plant to each part in order to minimize a chosen cost criterion. At a lower level, a path following logic computes the control inputs in order to follow the assigned path, while satisfying all constraints. The approach is tested here in simulations, reporting extremely good performance as measured by closed-loop cost function values and computational efficiency.

I. INTRODUCTION

Research in advanced manufacturing solutions is motivated today by the strong global competition, together with the combined trends of higher customization, more agile supply, and higher sustainability [1], [2]. This domain spans from industrial communications to collaborative robotics, from human-machine interaction to routing and logistics, leading to a large number of challenging problems [3], [4], [5]. Among the latter, we focus on the real-time control of components’ routing in discrete manufacturing plants. Depending on the specific manufacturing process at hand, this problem may entail several requirements. The discrete parts must be routed via physical lines that present handling constraints, for example in terms of limited movement speed. Different lines may also merge, leading to possible lockouts to be avoided. Moreover, the processing time at each station may be uncertain, and the sequence of jobs to be done on a part may be not fully known a priori, since it can change depending on the outcome of each job. Component unavailabilities or faults may also affect the production lines. Finally, sustainability goals translate to minimization of waste and of energy consumption. From a control engineering perspective, the mathematical transcription of this problem leads to a prohibitive large-scale integer or mixed-integer optimal control program, involving a

L. Fagiano, L. Cucas and R. Scattolini are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy.

M. Tanaskovic is with Singidunum University, 32 Danijelova St., Belgrade, 160622 Serbia.

A. Cataldo is with the Institute of Industrial Technology and Automation, National Research Council, Via Alfonso Corti 12, 20133 Milano, Italy.

Corresponding author: L. Fagiano, lorenzo.fagiano@polimi.it.

This research was funded by a grant from the Italian Ministry of Foreign Affairs and International Cooperation (MAECI), project “Real-time control and optimization for smart factories and advanced manufacturing”.

M. Tanaskovic was also supported by the Science Fund of the Republic of Serbia under the project DECIDE.

dynamical system with discrete state variables, discrete input commands, and discrete output measurements, and subject to temporal logic constraints and external disturbance signals, where the goal is to guarantee the required throughput with minimal waste and energy cost. To tame such a complexity, hierarchical approaches are adopted, involving rule-based techniques [6], [7], [8], integer programming [9], multi-agent architectures [10], short-term simulation and ordinal optimization [11], heuristic search combined with Petri nets [12], and model predictive control (MPC) [13], [14], [15], [16]. In particular, in [15] a receding horizon approach has been employed to control in real-time a de-manufacturing plant composed of 35 nodes. In an analogy with fluid modeling, this approach adopted a *Eulerian description*, where the state vector includes the status of each node in the plant (conceptually similar to a control volume in fluid dynamics). The resulting policy provides the optimal solution to the finite horizon routing problem at each time step, however with rather high computational cost.

To overcome this problem, this paper contributes two novelties: 1) a shift of perspective from a Eulerian to a *Lagrangian description*, where the system state includes the status of each part that must be routed, instead of each node; and 2) a hierarchical MPC structure, where the receding horizon strategy allocates a path to each part (as well as the part’s position on the path) and a lower-level logic computes the control inputs to follow such a path. We tested the new approach in simulation and report extremely good performance as measured by closed-loop cost function values and computational efficiency, also with very large prediction horizon values. These features pave the way to a number of subsequent research steps, which will culminate with the experimental testing on the pilot plant considered in [15].

II. EULERIAN SYSTEM MODEL AND PROBLEM DESCRIPTION

We consider a discrete manufacturing plant composed of a finite number $N_n \in \mathbb{N}$ of nodes. At each discrete time instant k , each node $h = 1, \dots, N_n$ may be empty or it may host one (and only one) part being processed. For a reference, consider the diagram of Fig. 1 representing the small-scale system that we use in this paper to test the proposed approach, composed of 12 nodes. A more complex diagram representing a laboratory testbed at the National Research Council in Milano can be found in [15].

The boolean $z_h(k) \in \{0, 1\}$ indicates whether a part is present at node h (i.e., $z_h(k) = 1$) or not. We assume that N_t out of N_n nodes are *transportation modules*, and the remaining $N_m = N_n - N_t$ are *machines*. In particular, let

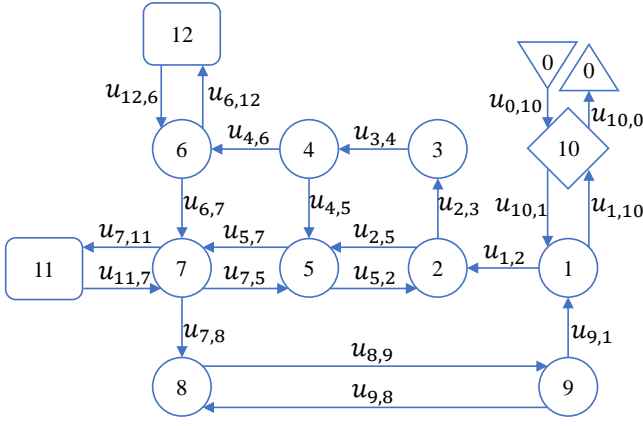


Fig. 1. Small-scale system considered as test case in this paper. Node 10 is both the loading and unloading one (i.e., $h_l = h_u = 10$) and nodes 11,12 are machines (i.e., $\mathcal{M} = \{11, 12\}$).

us denote the set of indexes of machine nodes as

$$\mathcal{M} = \{h : \text{node } h \text{ is a machine}\}.$$

Each node (be it a transportation module or a machine) is able to either hold one part in place, or to move it to a fixed number of specific, directly connected nodes according to the plant topology (see, e.g., Fig. 1). Each machine must, in addition, execute a specific job on each part it receives. Without loss of generality we assume that a movement from one node to a connected one lasts one time step (direct movement), while the job carried out by a machine m lasts an integer number $L_m \geq 1$ of time steps. The boolean control signal $u_{h,j}(k) \in \{0, 1\}$ dictates whether a part will move from node h at time k to node j at time $k + 1$. Finally, we also assume that two special nodes are present, the *loading* node and the *unloading* one, with indexes h_l and h_u , respectively. These nodes are the interface between the plant under study and the outside, denoted with index 0, through two control variables: u_{0,h_l} can move to the loading node a part from outside the plant, e.g. from a buffer containing the incoming parts that must be processed, while a part can be moved from the unloading node to the outside via the control variable $u_{h_u,0}$, e.g. to a buffer of finished parts. We denote with $N_f(k)$ the total number of finished parts at time k . In summary, the number N_u of boolean control signals to be computed at each time step is equal to the number of valid direct transitions among the nodes, plus the two loading and unloading commands $u_{0,h_l}, u_{h_u,0}$. We collect these inputs into a column vector, denoted with $U(k) \in \{0, 1\}^{N_u}$.

For each node $h = 1, \dots, N_n$, we define the following sets.

Definition 1: (Outgoing and Incoming sets)

- The *outgoing set* \mathcal{O}_h is the set containing the indexes of all nodes that can be reached directly from h , including possibly the outside, i.e. $\mathcal{O}_h = \{j : \exists u_{h,j}\}$;
- The *incoming set* \mathcal{I}_h is the set containing the indexes of all nodes for which h is a direct destination, including possibly the outside, i.e. $\mathcal{I}_h = \{j : \exists u_{j,h}\}$.

We thus have $\{0\} \in \mathcal{O}_{h_u}$ and $\{0\} \in \mathcal{I}_{h_l}$. Defining $z =$

$[z_1, \dots, z_{N_n}]^T$ (\cdot^T is the vector transpose operation) and

$$v(k) = \begin{bmatrix} \sum_{j \in \mathcal{I}_1} u_{j,1}(k) - \sum_{j \in \mathcal{O}_1} u_{1,j}(k) \\ \vdots \\ \sum_{j \in \mathcal{I}_{N_n}} u_{j,N_n}(k) - \sum_{j \in \mathcal{O}_{N_n}} u_{N_n,j}(k) \end{bmatrix}$$

we can introduce the following linear model describing the plant's behavior:

$$\begin{aligned} z(k+1) &= z(k) + v(k) \\ N_f(k+1) &= N_f(k) + u_{h_u,0}(k) \end{aligned} \quad (1)$$

This model corresponds to a Eulerian description of the system, where the nodes are taken as control volumes, the system state corresponds to the number of parts in each of these volumes, and the model essentially corresponds to a series of mass conservation equations. To keep consistency with the real system, the boolean control inputs must comply with the following operational constraints at all time steps:

$$\sum_{j \in \mathcal{O}_h} u_{h,j}(k) \leq 1, \quad h = 1, \dots, N_n \quad (2a)$$

$$\sum_{j \in \mathcal{I}_h} u_{h,j}(k) \leq 1, \quad h = 1, \dots, N_n \quad (2b)$$

$$\sum_{j \in \mathcal{O}_h} u_{h,j}(k) = 0, \quad \forall h : z_h(k) = 0 \quad (2c)$$

$$\sum_{j \in \mathcal{I}_h} u_{j,h}(k) = 0, \quad \forall h : z_h(k) = 1 \wedge \sum_{j \in \mathcal{O}_h} u_{h,j}(k) = 0 \quad (2d)$$

Constraints (2a)-(2b) impose that a part shall move to at most one destination from node h , and that only one part shall reach node h at the next time step. Constraint (2c) states that all control signals from an empty node shall be zero, finally constraint (2d) imposes that no part can move to node h if the latter is occupied and it will hold its current part in the next step.

Moreover, temporal logic constraints on the control inputs pertaining to machine nodes arise, due to the fact that once a job is started it must be completed before the part can be moved. Denoting with k_m the time when a new job is started by machine m , such constraints take the form:

$$\sum_{j \in \mathcal{O}_m} u_{m,j}(k) = 0, \quad \forall k \leq k_m + L_m, \forall m \in \mathcal{M} : z_m(k) = 1 \quad (3)$$

The problem we address can be described as follows: derive a control policy that computes, at each time instant k , all of the control variables $u_{h,j}$ in order to satisfy the operational constraints (2)-(3) and to minimize a suitably defined cost criterion.

In [15], this problem has been addressed resorting to MPC, after a suitable manipulation of the model and of the constraints that leads to a mixed logical dynamic (MLD) formulation and a large-scale mixed-integer linear program to be solved at each time step. The approach has been tested experimentally with good performance, however it leads to rather high computational complexity: as an example, in the test case of Fig. 1, 160 integer auxiliary variables need to be introduced per each time step in the prediction horizon (e.g., with a 5-time-steps horizon about 800 integer variables are

used).

The approach introduced in this paper, presented next, aims to overcome this issue by taking a different perspective on the problem at hand.

III. LAGRANGIAN SYSTEM MODEL AND PROPOSED APPROACH

To reduce the computational complexity while still retaining an optimization-based predictive approach, we propose here the hierarchical control structure presented in Fig. 2: a *low-level greedy path following strategy* is in charge

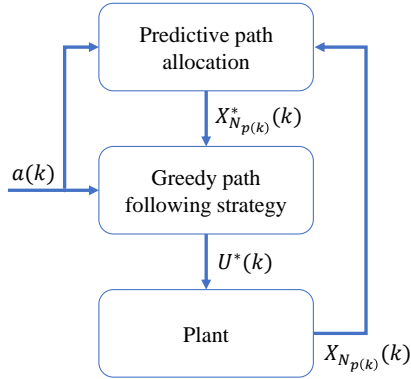


Fig. 2. Hierarchical control approach proposed in this paper.

to compute feasible control inputs and to move forward each part along its assigned path, and a *high-level model predictive path allocation* module (re-)assigns the paths to all parts in order to optimize a user-defined performance metric. As suggested by the adopted terminology, in this new approach we follow the parts' trajectories instead of keeping track of the status of each control volume, i.e. we adopt a Lagrangian description of the plant instead of an Eulerian one. Consistency/translation between the two descriptions is provided by the fact that each path is a sequence of nodes, thus the position of a part on a given path unequivocally identifies the node where that part is located.

A. Lagrangian model state

Let us denote with $i = 1, \dots, N_p(k)$ an index that identifies each one of the $N_p(k) \in \mathbb{N}$ parts in the plant at time k . The value of $N_p(k) \in \mathbb{N}$ can change over time as new parts enter the plant and/or finished ones exit. We further denote with $S = \{1, 2, \dots, N_s\}$ a set of integers, each one corresponding unequivocally to a *sequence* (or path). Such sequences are assumed to be precomputed and stored. For each $s \in S$, the operator $\mathcal{S}(s)$ returns the actual sequence corresponding to index s . Each sequence $\mathcal{S}(s)$ has the following structure:

$$\mathcal{S}(s) = \left\{ \begin{bmatrix} h_1 \\ g_1 \end{bmatrix}, \dots, \begin{bmatrix} h_p \\ g_p \end{bmatrix}, \dots, \begin{bmatrix} h_{N_s} \\ g_{N_s} \end{bmatrix} \right\} \quad (4)$$

where N_s is the sequence length, $p = 1, \dots, N_s$ is the position along the sequence, and h_p, g_p are integers corresponding to nodes in the plant. In particular, each value of h_p corresponds to a node that is either equal to h_{p+1} (i.e. the part shall be held), or directly connected to h_{p+1} (i.e., the part shall be moved from node h_p to h_{p+1}), while each

value of g_p is the index of a node chosen as goal for that part of the sequence. Usually, such goal indexes correspond to machines or to the outside (node 0).

We indicate with $s_i(k) \in S$ the sequence that part i is following at time k , with $p_i(k) \in \mathbb{N}$ the position of part i along such a sequence, and with $\mathcal{S}(s_i(k))^{(1,p_i(k))}, \mathcal{S}(s_i(k))^{(2,p_i(k))}$ the first and second entry, respectively, of the vector in position $p_i(k)$ of sequence $\mathcal{S}(s_i(k))$ (compare (4)). For example, referring to Fig. 1, the sequence identified by index $s = 1$ could correspond to $\mathcal{S}(1) = \left\{ \begin{bmatrix} 10 \\ 12 \end{bmatrix}, \begin{bmatrix} 1 \\ 12 \end{bmatrix}, \begin{bmatrix} 2 \\ 12 \end{bmatrix}, \begin{bmatrix} 3 \\ 12 \end{bmatrix}, \begin{bmatrix} 4 \\ 12 \end{bmatrix}, \begin{bmatrix} 6 \\ 12 \end{bmatrix}, \begin{bmatrix} 12 \\ 12 \end{bmatrix}, \begin{bmatrix} 6 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 \\ 0 \end{bmatrix}, \begin{bmatrix} 8 \\ 0 \end{bmatrix}, \begin{bmatrix} 9 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 \\ 0 \end{bmatrix} \right\}$, and a part i with $s_i(k) = 1$ and $p_i(k) = 3$ would be located at node $h = 2$ at time k , i.e. $\mathcal{S}(s_i(k))^{(1,p_i(k))} = 2$ and have as goal the machine node $\mathcal{S}(s_i(k))^{(2,p_i(k))} = 12$. Moreover, we denote with \underline{k}_i the time step when part i appeared on the plant, and with $t_i(k)$ the time elapsed since then:

$$t_i(k) = k - \underline{k}_i. \quad (5)$$

In our Lagrangian model the state of part i is given by:

$$\mathbf{x}_i(k) = \begin{bmatrix} s_i(k) \\ p_i(k) \\ t_i(k) \end{bmatrix}. \quad (6)$$

Finally, we denote with

$$r_i(k) = \text{card}(\mathcal{S}(s_i(k))) - p_i(k) \quad (7)$$

the number of remaining nodes that part i shall visit to complete its current sequence. For later use, we also collect all the state variables in vector

$$\mathbf{x}_{N_p(k)}(k) = [\mathbf{x}_1(k)^T, \dots, \mathbf{x}_{N_p(k)}(k)^T]^T \in \mathbb{N}^{3N_p}, \quad (8)$$

which represents the overall state of the Lagrangian plant model. Note that such a state vector can change dimension in time as it depends on the value of $N_p(k)$ (which we denote with the subscript $\cdot_{N_p(k)}$ in (8)). Albeit rather unusual in dynamical models, this feature does not lead to any technical problem as long as consistency with the Eulerian model is ensured. In turn, this is obtained by always applying feasible inputs to the plant, as achieved by our path following algorithm, introduced next.

B. Greedy path following strategy and closed-loop Lagrangian model

The greedy path following strategy is a rule-based controller that acts according to the following principles: a) if possible, move each part forward in its current sequence; b) if the next node in the sequence is blocked, wait; c) if a potential conflict among parts is detected, the part with smallest $r_i(k)$ value shall move, and the other ones shall wait. To account for new parts that must be loaded to the plant from the outside, we introduce the boolean $a(k)$, which is equal to 1 when such a new part is available to be moved to the loading node.

Algorithm 1 *Greedy path following strategy*. At each time step k :

1. Compute $r_i(k)$, $i = 1, \dots, N_p(k)$ according to (7);
2. Compute the one-step-ahead predicted states $\hat{\mathbf{x}}_i$, $i = 1, \dots, N_p(k)$, by forward-propagation of all parts along their current paths:

$$\begin{aligned} \hat{p}_i(k+1) &= p_i(k) + 1 \\ t_i(k+1) &= t_i(k) + 1 \\ \hat{\mathbf{x}}_i(k+1) &= \begin{bmatrix} s_i(k) \\ \hat{p}_i(k+1) \\ t_i(k+1) \end{bmatrix} \end{aligned} \quad (9)$$

3. For each node $h = 1, \dots, N_n$, compute the number of potential conflicts $n_h(k+1)$ as: $n_h(k+1) = \sum_{i=1}^{N_p(k)} c(\hat{\mathbf{x}}_i(k+1), h) - 1$, where $c(\hat{\mathbf{x}}_i(k+1), h) = \begin{cases} 1 & \text{if } \mathcal{S}(1, s_i(k))^{(\hat{p}_i(k+1))} = h \\ 0 & \text{otherwise} \end{cases}$;
4. If $n_h(k+1) = 0$ for all h , then go to step 5. Else, for each node $h : n_h(k+1) > 0$ do conflict resolution:

- 4.a. Compute the set containing the indexes of conflicting parts $\mathcal{C}_h(k) = \{i : c(\hat{\mathbf{x}}_i(k+1), h) = 1\}$;
- 4.b. Check if a part is being held at node h :

$$\begin{aligned} \bar{i}_h(k) &= \{i \in \mathcal{C}_h(k) : \mathcal{S}(\hat{s}_i(k+1))^{(1, \hat{p}_i(k+1))} \\ &= \mathcal{S}(s_i(k))^{(1, p_i(k))} = h\} \end{aligned}$$

- 4.c. Compute the set of parts that are most advanced in their own path: $\mathcal{L}_h(k) = \left\{ i : r_i(k) = \min_{l \in \mathcal{C}_h(k)} r_l(k) \right\}$
- 4.d. Compute the index $i_h^*(k)$ of the part with highest priority:
If $\bar{i}_h(k) \neq \emptyset$ then $i_h^*(k) = \bar{i}_h(k)$
Else if $\text{card}(\mathcal{L}_h(k)) = 1$ then $i_h^*(k) = \mathcal{L}_h(k)$
Else $i_h^*(k) = \arg \min_{l \in \mathcal{L}_h(k)} t_l(k)$.

- 4.e. $\forall i \in \mathcal{C}_h(k) : i \neq i_h^*(k)$, correct the corresponding one-step-ahead predicted state as: $\hat{p}_i(k+1) = p_i(k)$, $\hat{\mathbf{x}}_i(k+1) = \begin{bmatrix} s_i(k) \\ \hat{p}_i(k+1) \\ t_i(k+1) \end{bmatrix}$.
- 4.f. Go to 3.

5. Apply to the plant the following inputs, corresponding to the computed part movements:

$$\begin{aligned} \forall h, j : \exists u_{h,j} \wedge h \neq 0 \\ u_{h,j}(k) &= 1, \text{ if } \exists i : \mathcal{S}(s_i(k))^{(1, \hat{p}_i(k+1))} = j \\ &\quad \wedge \mathcal{S}(s_i(k))^{(1, p_i(k))} = h \\ u_{h,j}(k) &= 0, \text{ else.} \\ u_{0,h_l}(k) &= 1, \text{ if } a(k) = 1 \wedge \\ &\quad \nexists i : \mathcal{S}(s_i(k))^{(1, \hat{p}_i(k+1))} = h_l \\ u_{0,h_l}(k) &= 0, \text{ else.} \end{aligned}$$

Step 4.d. of **Algorithm 1** sets the priority as follows: a part being held at a node has the highest priority, if no part is held then the one that is most advanced in its own sequence (i.e. minimal $r_i(k)$) has the second-highest priority, if more than one part has minimal $r_i(k)$ then the one that has been in the plant for the longest time has the third-highest priority. The feedback control policy defined by **Algorithm 1**

corresponds to a set of functions κ_{N_p} , $N_p \in \mathbb{N}$. Each one of these functions pertains to a specific number of parts N_p and its input arguments are the corresponding Lagrangian state $X_{N_p(k)}$ and signal $a(k)$, while the output of all of them is a vector of plant commands $U \in \{0, 1\}^{N_u}$ (see step 5. of **Algorithm 1**):

$$U(k) = \kappa_{N_p(k)}(X_{N_p(k)}(k), a(k)). \quad (10)$$

Under mild assumptions on the employed sequences, one can prove that such a control policy generates inputs that are always feasible. However, input feasibility by itself does not prevent the controlled system from running into a lockout, and in general the greedy path following approach can give suboptimal behavior with respect to the performance criteria of interest. On the other hand, when combined with the model (1), **Algorithm 1** allows one to predict the system behavior without having to explicitly enforce the challenging constraints (2)-(3) and at extremely low computational cost. We exploit such closed-loop predictions in a high-level MPC strategy, described in the next Section.

Before proceeding further, we also introduce the closed loop Lagrangian model of the system, provided by the following algorithm.

Algorithm 2 *Closed-loop Lagrangian plant model.* At each time step k :

1. Run **Algorithm 1** with the current values of $X_{N_p(k)}(k)$ and $a(k)$ as inputs, collect all the resulting values of $\hat{\mathbf{x}}_i(k+1)$ and $u_{h,j}(k)$, $\forall (h, j) : \exists u_{h,j}$;
2. Compute the Lagrangian state dimension $N_p(k+1)$ as $N_p(k+1) = N_p(k) + u_{0,h_l} - u_{h_u,0}$;
3. If $u_{0,h_l} = 1$, generate the state $\tilde{\mathbf{x}}(k+1)$ of the new part that will be loaded to the plant at time $t+1$;
4. For all $i : \mathcal{S}(s_i(k))^{(1, \hat{p}_i(k+1))} \neq h_l$, compute the state $\mathbf{x}_i(k+1) = \hat{\mathbf{x}}_i(k+1)$.
5. Compute the Lagrangian state $X_{N_p(k+1)}(k+1)$ by stacking all vectors $\mathbf{x}_i(k+1)$ computed at step 4. and, if available, vector $\tilde{\mathbf{x}}(k+1)$ computed at step 3. Set $k = k+1$ and go to 1.

The state initialization of a new part at step 3. can be done by assigning a sequence $s \in S$ and position p to it. Since their state value is not updated at step 4., parts that are unloaded from the plant naturally disappear from the Lagrangian model.

Similarly to the control policy (10), the system model defined by **Algorithm 2** also corresponds to a set of functions, $f_{(N_p^+, N_p)} : 3\mathbb{N}^{N_p} \rightarrow 3\mathbb{N}^{N_p^+}$, each one pertaining to a specific pair of part quantities, i.e. those at the current and at next time steps, while the signal $a(k)$ is an exogenous input:

$$X_{N_p(k+1)}(k+1) = f_{(N_p(k+1), N_p(k))}(X_{N_p(k)}(k), a(k)). \quad (11)$$

C. Model predictive path allocation

At each time step, the predictive control logic chooses whether to keep each part on its current path $s_i(k)$ and at its current position $p_i(k)$, or to change one or both of these elements in order to optimize the predicted plant performance. The result is a dynamic, optimization-based path allocation strategy that can exploit very large prediction horizon values, thus guaranteeing the absence of lockouts,

and allows one to easily consider different performance indexes and to generally improve the plant behavior with respect to the one obtained by the greedy path following policy alone.

At each time step k , let us consider the following sets $\mathcal{X}_i(k)$, $i = 1, \dots, N_p(k)$:

$$\text{if } \mathcal{S}(s_i(k))^{(1,p_i(k))} \notin \mathcal{M} : \\ \mathcal{X}_i(k) = \left\{ \begin{array}{l} (s, p) \in S \times \mathbb{N} : \\ \mathcal{S}(s)^{(1,p)} = \mathcal{S}(s_i(k))^{(1,p_i(k))} \\ \wedge \mathcal{S}(s)^{(2,p)} = \mathcal{S}(s_i(k))^{(2,p_i(k))} \end{array} \right\} \quad (12a)$$

$$\text{else if } \mathcal{S}(s_i(k))^{(1,p_i(k))} \in \mathcal{M} : \\ \mathcal{X}_i(k) = \left\{ \begin{array}{l} (s, p) \in S \times \mathbb{N} : \\ \mathcal{S}(s)^{(1,p-j)} = \mathcal{S}(s_i(k))^{(1,p_i(k)-j)}, \\ j = 0, \dots, k - k_{\mathcal{S}(s_i(k))^{(1,p_i(k))}} \\ \wedge \mathcal{S}(s)^{(2,p)} = \mathcal{S}(s_i(k))^{(2,p_i(k))} \end{array} \right\} \quad (12b)$$

where $k_{\mathcal{S}(s_i(k))^{(1,p_i(k))}}$ is the time step when part i started the job of machine $m = \mathcal{S}(s_i(k))^{(1,p_i(k))}$ (compare (3)). Namely, each set $\mathcal{X}_i(k)$ contains all the pairs (s, p) of sequence index and position index such that the corresponding vector $[\mathcal{S}(s)^{(1,p)} \mathcal{S}(s)^{(2,p)}]^T$ corresponds to that of part i at time k , also considering a possible ongoing job and its remaining duration, if $\mathcal{S}(s_i(k))^{(1,p_i(k))}$ is a machine node. These sets are never empty by construction, since they always include the current pair $(s_i(k), p_i(k))$. At any time k , exchanging these two components of the state $\mathbf{x}_i(k)$ to any other pair $(s, p) \in \mathcal{X}_i(k)$ implies that we are allocating to part i another sequence and/or position among those that are compatible with its current physical location and goal. Our high-level predictive controller exploits precisely this feature, as described in the following algorithm. We denote with $X_{N_p(o|k)}(o|k)$, $\mathbf{x}(o|k)$ the predictions of plant input and Lagrangian states, respectively, computed at time k and pertaining to time $k + o$.

Algorithm 3 Model Predictive Path Allocation.

1. At time k acquire the state variables $\mathbf{x}_i(k)$, $i = 1, \dots, N_p(k)$ and compute the corresponding sets $\mathcal{X}_i(k)$;
2. Solve the following finite horizon optimal control problem (FHOCP):

$$\min_{(\sigma_i, \pi_i), i=1, \dots, N_p(k)} \sum_{o=0}^N \ell_{N_p(o|k)}(X_{N_p(o|k)}(o|k)) \quad (13a)$$

subject to

$$\mathbf{x}_i(0|k) = [\sigma_i, \pi_i, t_i(k)]^T, \quad i = 1, \dots, N_p(k) \quad (13b)$$

$$X_{N_p(o|k)}(0|k) = [\mathbf{x}_1(0|k)^T, \dots, \mathbf{x}_{N_p(k)}(0|k)^T]^T \quad (13c)$$

$$X_{N_p(o+1|k)}(o+1|k) = f_{(N_p(o+1|k), N_p(o|k))}(X_{N_p(o|k)}(k), a(o|k)), \quad (13d) \\ o = 0, \dots, N - 1$$

$$(\sigma_i, \pi_i) \in \mathcal{X}_i(k), \quad i = 1, \dots, N_p(k) \quad (13e)$$

where $N \in \mathbb{N}$ is the prediction horizon, the sequence $a(o|k) \in \{0, 1\}$, $o = 0, \dots, N - 1$ contains the

predictions of new parts that need to be worked (if available), and the stage cost functions $\ell_{N_p}(X_{N_p})$ are chosen by the designer according to the plant performance indicator of interest.

3. Let (σ_i^*, π_i^*) , $i = 1, \dots, N_p(k)$ be the solution to (13). Compute the new state vectors $\mathbf{x}_i^*(k)$ as:

$$\mathbf{x}_i^*(k) = \begin{bmatrix} \sigma_i^* \\ \pi_i^* \\ t_i(k) \end{bmatrix}, \quad i = 1, \dots, N_p(k) \\ X_{N_p(k)}^*(k) = [\mathbf{x}_1^*(k)^T, \dots, \mathbf{x}_{N_p(k)}^*(k)^T]^T$$

and provide these values to **Algorithm 1** to compute the control inputs via (10):

$$U^*(k) = \kappa_{N_p(k)}(X_{N_p(k)}^*(k), a(k)).$$

4. Apply to the plant the control inputs $U^*(k)$, set $k = k + 1$, go to **1.**

Note that the optimization variables (σ_i, π_i) , $i = 1, \dots, N_p(k)$ pertain only to the current time step, i.e. the sequence index is not changed during the predictions. This results in possible sub-optimality but increased computational efficiency, similarly to what is done in move blocking strategies in MPC, see e.g. [17]. On the other hand, being a receding horizon strategy, **Algorithm 3** is able to change the sequence and position indexes $s_i(k), p_i(k)$ of all states at each time step k , resulting in practice in good closed-loop performance. Signal $a(k)$, which is managed by the greedy path following algorithm as described in Section III-B, is considered as an external disturbance, of which a prediction may be available (otherwise one can simply set $a(o|k) = 0$ in (13d)).

IV. NUMERICAL RESULTS

We present the tests of the hierarchical approach on the small-scale example of Fig. 1, with $N_u = 22$ boolean control inputs. The two machine nodes 11, 12 have the same processing time $L_{11} = L_{12} = 3$ time steps, and each part must visit first machine 12, then machine 11 before leaving the plant from node 10, which is also the loading node. We assume that $a(k) = 1 \forall k$, i.e. a new part is loaded to the plant whenever the loading node is free, and that the predictive controller does not have this information. The maximum throughput of the plant depends on the processing time of machine 12 and on the fact that node 10 has to switch between loading a new part or unloading a finished one, thus adding two additional time steps. Its value is thus equal to $1/(L_{12} + 2) = 0.20$ parts per time step. We set a prediction horizon of $N = 50$ time steps, and we use as stage cost in (13a) the following function:

$$\ell_{N_p(o|k)} = \sum_{i=1}^{N_p(o|k)} r_i(o|k) + \beta \sum_{i=1}^{N_p(o|k)} \sum_{i=1}^{n_u} U(o|k) \quad (14)$$

where $\beta \geq 0$ is a weighting factor, $r_i(o|k)$ is computed as in (7) considering the predicted Lagrangian states, and $U(o|k) = \kappa_{N_p(o|k)}(X_{N_p(o|k)}^*(o|k), 0)$ is the vector of simulated actuation commands given to the plant. As regards the sequence computation, since the considered example is essentially a series manufacturing process, we adopt here

a single path, composed of redundant sub-sequences going several times through all possible loops across nodes 2, 3, 4, 5, 6, 7 (see Fig. 1) and of sub-sequences of identical values for each node, in order to provide the predictive controller with the option to make one part wait in place by shifting it back with such sub-sequences. We ran all

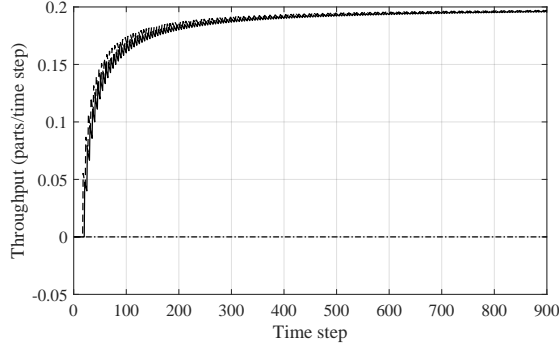


Fig. 3. Simulation example. Course of the plant throughput expressed as number of finished parts per time step, with $\beta = 5$ (dashed line), $\beta = 6$ (solid), and $\beta = 8$ (dash-dotted).

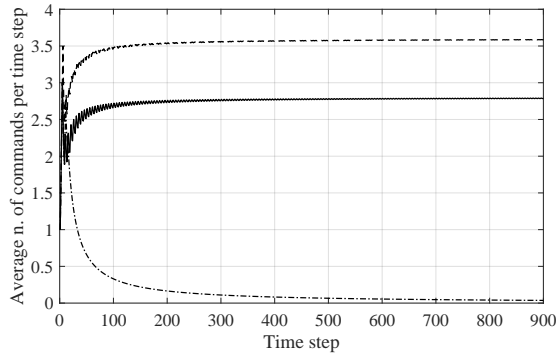


Fig. 4. Simulation example. Course of the average number of input commands per time step, with $\beta = 5$ (dashed line), $\beta = 6$ (solid), and $\beta = 8$ (dash-dotted).

the simulations starting from one part in node 10. In this example, the maximum throughput can be reached with different strategies that lead to different values of energy consumption: in fact, during each job it is possible to let the waiting parts be held on the nodes, or to make them circulate in the available loops within the plant. We illustrate that the proposed strategy switches between these two behaviors as the value of β decreases. This is clearly visible in Figs. 3-4: with $\beta = 5$ the plant reaches the maximum throughput and a number of commands per time step equal to 3.5, while with $\beta = 6$ the same throughput is obtained with only 2.5 commands per time step, i.e. 30% less. In both cases, a number of parts oscillating between 7 and 8 is present on the plant at each time step, after the initial transient. If we further increase β , the controller reaches a lockout with eight parts on the plant, since it becomes more convenient to avoid any actuation rather than to push the parts forward in their paths. This is also shown in Figs. 3-4.

Finally, regarding the computational aspects, we solved the problem (13) via extensive search over all possible valid (σ_i, π_i) pairs. On a Laptop with 8GB RAM and an Intel Core i7 CPU at 2.6 GHz running Matlab, the resulting computational time is 0.45 s per time step, without any attempt to improve the solver efficiency (e.g. by parallelizing the computations and/or adopting a non-brute-force approach to solve the optimization problem).

REFERENCES

- [1] European Commission, "Factory of the future," *Multi-Annual Roadmap for the Contractual PPP Under Horizon 2020*. Belgium: Publications Office of the European Union, 2013.
- [2] P. Zheng, H. Wang, Z. Sang, R. Y. Zhong, Y. Liu, C. Liu, K. Mubarak, S. Yu, and X. Xu, "Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives," *Frontiers of Mechanical Engineering*, vol. 13, pp. 137–150, 2018.
- [3] W. Na, Y. Lee, N. Dao, D. N. Vu, A. Masood, and S. Cho, "Directional link scheduling for real-time data processing in smart manufacturing system," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3661–3671, Oct 2018.
- [4] S. Ren, Y. Zhang, Y. Liu, T. Sakao, D. Huisingsh, and C. M. Almeida, "A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions," *Journal of Cleaner Production*, vol. 210, pp. 1343 – 1365, 2019.
- [5] H. Yan, Q. Hua, Y. Wang, W. Wei, and M. Imran, "Cloud robotics in smart manufacturing environments: Challenges and countermeasures," *Computers & Electrical Engineering*, vol. 63, pp. 56 – 65, 2017.
- [6] Y. P. GUPTA, M. C. GUPTA, and C. R. BECTOR, "A review of scheduling rules in flexible manufacturing systems," *International Journal of Computer Integrated Manufacturing*, vol. 2, no. 6, pp. 356–377, 1989.
- [7] C. Saygin, F. Chen, and J. Singh, "Real-time manipulation of alternative routings in flexible manufacturing systems: A simulation study," *The International Journal of Advanced Manufacturing Technology*, vol. 18, pp. 755–763, 2001.
- [8] R. Bucki, B. Chramcov, and P. Suchánek, "Heuristic algorithms for manufacturing and replacement strategies of the production system," *Journal of Universal Computer Science*, vol. 21, no. 4, pp. 503–525, apr 2015.
- [9] S. K. Das and P. Nagendra, "Selection of routes in a flexible manufacturing facility," *International Journal of Production Economics*, vol. 48, no. 3, pp. 237 – 247, 1997.
- [10] K. Kouiss, H. Pierreval, and N. Mebarki, "Using multi-agent architecture in fms for dynamic scheduling," *Journal of Intelligent Manufacturing*, vol. 8, pp. 41–47, 1997.
- [11] C. Peng and F. Chen, "Real-time control and scheduling of flexible manufacturing systems: An ordinal optimisation based approach," *The International Journal of Advanced Manufacturing Technology*, vol. 14, pp. 775–786, 1998.
- [12] A. R. Moro, H. Yu, and G. Kelleher, "Hybrid heuristic search for the scheduling of flexible manufacturing systems using petri nets," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 240–245, 2002.
- [13] F. D. Vargas-Villamil and D. E. Rivera, "Multilayer optimization and scheduling using model predictive control: application to reentrant semiconductor manufacturing lines," *Computers & Chemical Engineering*, vol. 24, no. 8, pp. 2009 – 2021, 2000.
- [14] —, "A model predictive control approach for real-time optimization of reentrant manufacturing lines," *Computers in Industry*, vol. 45, no. 1, pp. 45 – 57, 2001.
- [15] A. Cataldo and R. Scattolini, "Dynamic pallet routing in a manufacturing transport line with model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1812–1819, Sep. 2016.
- [16] A. Cataldo, M. Morescalchi, and R. Scattolini, "Fault tolerant model predictive control of a de-manufacturing plant," *The International Journal of Advanced Manufacturing Technology*, vol. 9, no. 12, pp. 4803–4812, 2019.
- [17] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, no. 6, pp. 563 – 570, 2007.