Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 5655–5660

Automatic Routing Reconfiguration for Fault Tolerance in Smart Manufacturing Plants ⋆

Sonia De Santis * **Roberto Boffadossi** **,*** **Lorenzo Fagiano** **

* *MECO Research Team, Department of Mechanical Engineering, KU Leuven, Belgium (e-mail: sonia.desantis@kuleuven.be) and Flanders Make@KU Leuven, Belgium*

** *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy (e-mails: roberto.boffadossi@polimi.it, lorenzo.fagiano@polimi.it)*

*** *Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing (STIIMA), National Research Council, Milano, Italy*

Abstract: This paper focuses on the parts routing problem in a reconfigurable manufacturing plant, in presence of potential faults and uncertainty on the job scheduling and duration. The plant is modeled as a directed graph, where the nodes represent either transportation modules or machines, and the edges represent the allowed transitions between adjacent nodes. The parts move across the plant along predefined sequences of nodes, therefore the system state tracks the progress of the parts along such sequences and the control inputs are the transitions to be activated to command the parts movement. Provided the sequences, the proposed method automatically generates feedback control rules for deadlock avoidance, which are employed by a path following strategy to compute the suitable control inputs, complying with given temporal-logic constraints and avoiding deadlock states. Additionally, the approach is extended to deal with faults affecting the transportation modules via the selection of new feasible sequences and the online reconfiguration of the system state. Finally, the proposed approach is tested in high-fidelity simulations, showing high computational efficiency and throughput.

Keywords: Flexible and reconfigurable manufacturing systems, Smart manufacturing, Manufacturing plant control, Industry 4.0, Advanced manufacturing

1. INTRODUCTION

Smart manufacturing marks an important turning point in the industrial paradigm, as it enables to achieve better performance thanks to advanced technologies, such as artificial intelligence, data analysis, collaborative robotics, etc. (Meindl et al. (2021)). A key feature of smart manufacturing plants is the routing flexibility, which increases the process adaptability and allows to overcome the fault occurrence by suitable reconfigurations of the parts paths. However, such a flexibility also introduces a higher system complexity, as it requires advanced plant control methods, (Li and Si (2017); Ali and Wadhwa (2010); Morel et al. (2007)). In fact, without proper management, the parts can end up in plant deadlocks, i.e., stall situations due to conflicting movements of two or more parts in a loop.

Several solutions can be found in the literature for the problem at hand, considering different models of the plant (Yadav and Jayswal (2018)), such as automata (Silva et al. (2011)), Petri nets (Xiong et al. (1996)) and directed graphs (Cataldo and Scattolini (2016); Cataldo et al. (2019); Fagiano et al. (2020); Boffadossi et al. (2021)). In particular, in Fagiano et al. (2020), a Hierarchical Predictive Routing Control strategy (HPRC) was applied, featuring a low-level path following strategy and a high-level predictive path allocation. In Boffadossi et al. (2021), several improvements to the HPRC approach were presented, including the introduction of plant-specific handling constraints, a deadlock detection routine and a novel search tree exploration method. However, in these previous contributions, the deadlock avoidance required on-line tree search and backtracking procedures, which may be time consuming, and no handling of faults was considered.

To overcome these issues, this paper presents a new control approach for a reconfigurable plant modeled as a directed graph, featuring the automatic generation of deadlock avoidance rules, based on the theoretical findings of Zhang et al. (2006), and a fault recovery procedure.

After describing the problem and the approach in Sections 2 and 3, respectively, in Section 4 the method is showcased on a high-fidelity simulator, while Section 5 provides conclusions and future research directions.

⋆ This research was partially 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", and partially supported by the Flanders Make SBO project ARENA: Agile and Reliable Navigation.

10.1016/j.ifacol.2023.10.489

2. PROBLEM FORMULATION

2.1 Plant description

The considered system is a discrete manufacturing plant, composed of several machines connected by a modular transport line providing multiple routing options. For the sake of problem description, an example is shown in Fig. 1.

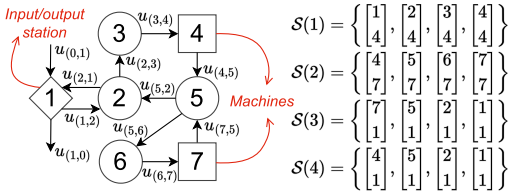


Fig. 1. Plant Graph representing a manufacturing plant.

The plant is modeled as a directed graph $G(N, E)$, named Plant Graph, where:

- The set N contains a finite number N_n of nodes representing the positions on the plant, of which N_t are *transportation nodes*, while $N_m = N_n - N_t$ are *machine nodes*: $\mathcal{M} = \{m : \text{node } m \text{ is a machine}\}$. A machine can be either a processing or measurement unit executing a specific task, or an input/output station. The subsets $\mathcal{M}_u, \mathcal{M}_{I/O} \subseteq \mathcal{M}$ identify the machines with uncertain outcome and the input/output stations, respectively.
- The set of directed edges E identifies a finite number N_u of possible movements between two adjacent nodes on the Plant Graph. An edge is indicated as an ordered pair (n_a, n_b), $n_a, n_b \in N$, where n_a is the start node and n_b is the arrival node.

Each one of the $N_p(k) \in \mathbb{N}$ parts present in the plant at time step k, denoted by an index $i = 1, ..., N_p(k)$, is directed to one target machine depending on its next scheduled job. After the job execution, the part moves towards the next target, until it is completely processed and thus can exit the plant, by targeting one of the input/output stations. The sequence of jobs can be uncertain, since the next job may depend on the outcome of the previous ones.

The motion between the machines is dictated by predefined paths (or sequences), identified by N_s integers and collected in set $S = \{1, \dots, N_s\}$. The paths can be selected freely by the designer, as long as they provide a connection between the machines to cover all the possible series of jobs. For each $s \in S$, the operator $\mathcal{S}(s)$ returns the corresponding sequence, composed of an ordered set of M_s vectors:

$$\mathcal{S}(s) = \left\{ \begin{bmatrix} n_{s,1} \\ g_{s,1} \end{bmatrix}, \cdots, \begin{bmatrix} n_{s,p} \\ g_{s,p} \end{bmatrix}, \cdots, \begin{bmatrix} n_{s,M_s} \\ g_{s,M_s} \end{bmatrix} \right\} \tag{1}$$

where, for each $p = 1, \dots, M_s$, $n_{s,p}$ indicates a node in the sequence s and $g_{s,p}$ indicates the target machine. The sequence assigned to part i at time step k is indicated as $s_i(k) \in S$, the position along such a sequence as $p_i(k) \in \mathbb{N}$, and the two operators $\mathcal{S}(s_i(k))^{(1,p_i(k))}$, $\mathcal{S}(s_i(k))^{(2,p_i(k))}$ return the first and the second entry of the vector in position $p_i(k)$ of sequence $\mathcal{S}(s_i(k))$, respectively. As an example, considering the Plant Graph in Fig. 1, $s_i(k) = 3$ and $p_i(k) = 2$ implies that part i occupies node 5 at time k, i.e., $\mathcal{S}(s_i(k))^{(1,p_i(k))} = 5$, and its target is the machine node 1, $\mathcal{S}(s_i(k))^{(2,p_i(k))} = 1$.

With this formalism, the state of part i at time step k consists of the two elements:

$$x_i(k) = [s_i(k), p_i(k)]^T. \tag{2}$$

The state of the plant is the collection of the parts states:

$$X(k) = [x_1(k)^T, ..., x_{N_p(k)}(k)^T]^T \in \mathbb{N}^{2N_p(k)}. \tag{3}$$

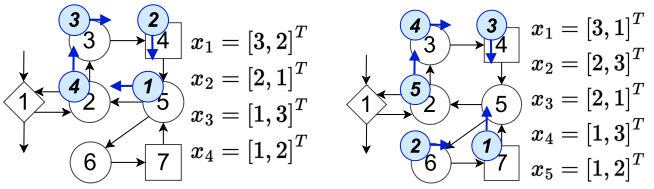
As mentioned above, the outcome of one or more jobs may be uncertain, so that the sequences followed by each part are assigned online after the execution of each uncertain job. The duration of each job might be uncertain as well, and it is modeled by a Boolean machine indicator r_m, set to 1 when machine m has finished its job. Moreover, plant-specific, state-dependent limitations on the allowed movements can also be present, giving rise to additional constraints on the plant state.

Provided the plant state, a path following strategy is employed to determine the set of Boolean control inputs $U(k) \in \{0, 1\}^{N_u}$, with the goal of propagating forward the motion of each part along its sequence, compatibly with all the mentioned operational constraints. Specifically, each variable $u_{(n_a,n_b)}(k) \in U(k)$, where $(n_a, n_b) \in E$, is associated with one edge on the Plant Graph and its value determines whether the motion along the corresponding edge is commanded: as an example, input $u_{(n_a,n_b)}(k) = 1$ if the movement along the transition (n_a, n_b) is commanded, such that the part occupying node n_a at time k moves to node n_b at time $k + 1$. Moreover, two control inputs for each input/output station $m \in \mathcal{M}_{I/O}$ describe the loading and unloading operations, and are indicated as $u_{(0,m)}$ and $u_{(m,0)}$, respectively. Finally, an exogenous Boolean variable $a(k)$ indicates the availability of a new part to enter the plant.

2.2 Deadlock description

Deadlocks are a common issue in flexible plants, since various parts share the same resources (transport line, buffers, machines, etc.). They are determined by circular wait situations: each part involved cannot move because it is waiting for other parts to move, including itself.

Specifically, two kinds of deadlock exist: primary and impending deadlocks. Primary deadlocks occur when two or more parts form a cycle on the Plant Graph while they are moving, thus they are indefinitely waiting for the next node along their sequence to be emptied, as an example see Fig. 2a. Impending deadlocks occur in specific



(a) Primary deadlock. (b) Impending deadlock.

Fig. 2. Deadlock examples in the plant depicted in Fig. 1. The parts are represented as blue circles on the top left of the occupied nodes, reporting the part index, and the corresponding blue arrows indicate the intended movement.

situations where the parts are still able to move, but their movement will inevitably lead to a primary deadlock in a finite number of time steps. For instance, Fig. 2b shows a plant state where part 1 or 3 can move towards plant node 5, but both these movements lead to a primary deadlock.

Ultimately, either kind of deadlocks are highly undesired, as they lead to the blockage of part or of the whole plant.

2.3 Problem formulation

The considered problem is to design a control strategy that is able to deal with all the following aspects:

1. The inputs must be consistent with the plant state, to apply a suitable control action and to avoid conflicts;
2. The uncertainty in the jobs duration and sequence, as described above, shall be dealt with;
3. Deadlock states must be avoided;
4. Faults preventing the activation of one of the N_u control inputs shall be accommodated.

3. PROPOSED ROUTING METHOD

The proposed approach is described in Fig. 3. Looking at the figure from left to right, at first a suitable set of paths is selected. Then, a set of routing rules to avoid deadlocks is automatically generated, based on the selected paths. These rules are employed by a path following strategy to move forward the parts along their assigned paths while ensuring that no conflict, deadlock or plant-specific constraint violation occurs.

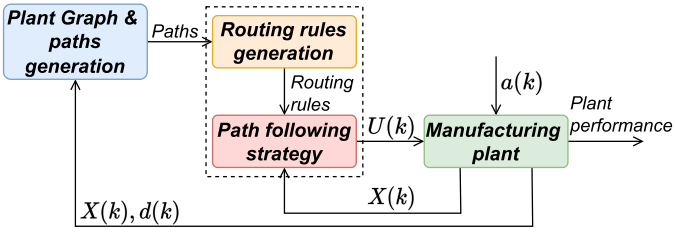


Fig. 3. Scheme of the proposed control strategy.

The paths generation is carried out before the process starts, then, in normal operating conditions, only the inner feedback loop remains active, until a potential fault is detected, indicated by the Boolean $d(k) = 1$. Then, the outer loop is activated, leading to the selection of new sequences and the reconfiguration of the plant state according to a fault recovery procedure. In the next subsections, each part of the strategy is described in detail.

3.1 Selection of sequences

The sequences are selected in such a way that the proper connections between the machines exist to cover all the possible jobs series. As an example, each sequence in Fig. 1 is chosen based on the process described in the following. Once a part is loaded in node 1, it first undergoes a job with uncertain outcome in machine node 4 (sequence 1). Afterwards, based on the job outcome, the part can exit the plant (sequence 4), or can be processed by the machine in node 7 (sequence 2). In the latter case, after the job in machine node 7, the part can exit the plant (sequence 3). In general, the choice of the sequences depends on various aspects. One could pick the shortest sequences to minimize the parts movements, or longer ones in case a relatively large number of parts are processed concurrently, making use of the additional nodes as buffers. In general, it is advisable to avoid sequences with many overlapping tracts in opposite directions, to reduce the risk of deadlock.

3.2 Automatic deadlock avoidance rules

Since deadlocks occur not only based on which plant nodes the parts occupy, but also on which ones they are headed to, a new directed graph, named the Transitions Graph, is created to better characterize the parts motion and derive the deadlock avoidance rules. The derivation of these rules is illustrated in the following, also referring to the plant in Fig. 1, with Transitions Graph shown in Fig. 4.

Provided the Plant Graph and the selected sequences, the Transitions Graph $G_T(N_T, E_T)$ is built in such a way that:

- Each node in the set N_T corresponds to a possible state that a part can assume, which in turn determines the current plant node and the next one, mapping into a transition between the two plant nodes, hence the name transition nodes. The latter are labelled $n_a.n_b$, where n_a is the starting plant node and n_b is the arrival one on the Plant Graph. In our example, the part state $x_i = [2, 1]^T$ corresponds to the transition node 4.5, because part i is occupying plant node 4 and is directed to 5. Note that, if two different states prescribe the movement along the same edge, they are mapped into the same transition node.
- Edges in E_T are drawn between every pair of transition nodes $n_a.n_b$ and $n_b.n_c$ with matching arrival and starting plant nodes. For instance, transition node 4.5 in Fig. 4 has to be connected to all the other ones with starting plant node 5: therefore, the outgoing edges from 4.5 connect nodes 5.2 and 5.6.

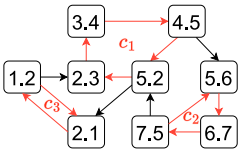


Fig. 4. Transitions Graph of the plant depicted in Fig. 1.

Once G_T is built, the next step is computing its cycles $c_j \in C$, which in our example are:

$$C = \{[2.3,\ 3.4,\ 4.5,\ 5.2],\ [5.6,\ 6.7,\ 7.5],\ [1.2,\ 2.1],\ [1.2,\ 2.3,\ 3.4,\ 4.5,\ 5.2,\ 2.1],\ [2.3,\ 3.4,\ 4.5,\ 5.6,\ 6.7,\ 7.5,\ 5.2],\ [1.2,\ 2.3,\ 3.4,\ 4.5,\ 5.6,\ 6.7,\ 7.5,\ 5.2,\ 2.1]\}. \quad (4)$$

Each cycle c_j maps into one or more plant states composed of n_{c_j} parts, where n_{c_j} is the number of nodes in c_j. For instance, cycle $c_3 = [1.2, 2.1]$ corresponds to a state where $n_{c_3} = 2$ parts form a cycle, one of them placed on plant node 1 and heading to 2, while the second one placed on plant node 2 and heading to 1. Note that this configuration can be prescribed by two different states: $x_1 = [1, 1]^T, x_2 = [3, 3]^T$ and $x_1 = [1, 1]^T, x_2 = [4, 3]^T$. However, some cycles do not correspond to realistic configurations, since they entail the simultaneous presence of more parts on the same plant node. As an example,

$c_4 = [1.2, 2.3, 3.4, 4.5, 5.2, 2.1]$ prescribes that two parts concurrently occupy plant node 2, since it contains nodes 2.3 and 2.1. Therefore, among the cycles in C, a subset C_P is selected, such that each $c_j \in C_P$ describes a feasible state configuration which is directly related to a primary deadlock. Denoting with $n_a^i.n_b^i$ the i^{th} transition node in the generic cycle c_j, we thus have:

$$C_P = \{c_j \in C : n_a^i \neq n_a^\ell \; \forall i, \ell = 1, \ldots, n_{c_j}\}. \quad (5)$$

In our example, $C_P = \{c_1, c_2, c_3\}$. Note that these also correspond to minimal cycles, i.e., without any sub-cycle. At this point, an additional set D is computed, whose elements d_j contain the set of plant nodes involved in each cycle $c_j \in C_P$. In the considered example, $D = \{[2, 3, 4, 5], [5, 6, 7], [1, 2]\}$. The latter set will be used later on, to determine impending deadlock states and to apply deadlock avoidance rules.

Regarding the identification of impending deadlock configurations, we first introduce the concept of connected cycles. Two cycles c_1 and c_2 are connected, indicated as $c_1 \leftrightarrow c_2$, if both these two conditions hold:

1. They contain one node with the same starting plant node, meaning that the intersection of the corresponding elements in D, $d_1 \cap d_2$, contains one node;
2. There exist sequences traversing an edge from a node in cycle c_1 to a node in c_2, and vice versa, with common plant node being $o = d_1 \cap d_2$.

For instance, considering c_1 and c_2 in our example, condition (1.) holds, since $d_1 \cap d_2 = 5$, and condition (2.) holds as well, since sequence 2, on position $1, 2, 3$ (see Fig. 1), prescribes the transition in c_1 with arrival plant node 5, $4.5 \in c_1$, immediately followed by the transition in c_2 with starting plant node 5, $5.6 \in c_2$, and sequence 3, on position $1, 2, 3$, prescribes the transition in c_2 with arrival plant node 5, $7.5 \in c_2$, immediately followed by the transition in c_1 with starting plant node 5, $5.2 \in c_1$.

Finally, n cycles are connected if they are such that $c_1 \leftrightarrow c_2 \leftrightarrow \ldots \leftrightarrow c_n$. In our case, it can be observed that $c_2 \leftrightarrow c_1$ and $c_1 \leftrightarrow c_3$, therefore $c_2 \leftrightarrow c_1 \leftrightarrow c_3$.

In summary, once the set of cycles C_P and D are computed, it is possible to assess whether they are connected and build the set I containing all the found connections, where each element $I_h \in I$ is a set of connected cycles:

$$I_h = \left\{ \{c_j\}_{j_1}^{j_\ell} \; : \; c_j \in C_P \forall j \; \wedge c_{j_1} \leftrightarrow \ldots \leftrightarrow c_{j_\ell} \right\} \quad (6)$$

where $\{c_j\}_{j_1}^{j_\ell}$ denotes a sequence of cycles. The number of elements in each set I_h is denoted by n_{I_h}. Note that the order of the n_{I_h} elements in set $I_h \in I$ does not matter, and if a set I_h contains more than two cycles, also the subsets containing the pairs of connected cycles in I_h must be added to I. In our example, $I = \{I_1, I_2, I_3\}$, $I_1 = \{1, 2\}$, $I_2 = \{1, 3\}$, $I_2 = \{1, 2, 3\}$.

In the following, it will be explained how these sets are used to derive deadlock avoidance rules.

Rules for deadlock avoidance The next result is derived based on the theoretical findings of Zhang et al. (2006).

Proposition 1. At any plant state, the following two conditions must hold to avoid primary and impending deadlocks, respectively:

1. The number of parts assuming the states in each cycle $c_j \in C_P$ must be smaller or equal than m_{c_j}, given by:
$$m_{c_j} = n_{c_j} - 1; \quad (7)$$
2. The number of parts assuming the states in each set of connected cycles $I_h \in I$ must be smaller or equal than m_{I_h}, given by:
$$m_{I_h} = |\cup_{j:c_j \in I_h} d_j| - n_{I_h} \quad (8)$$
where $|\cup_{j:c_j \in I_h} d_j|$ is the number of plant nodes appearing in at least one of the connected cycles in I_h.

Provided the sets C_P and I, it is possible to compute m_{c_j} for each cycle $c_j \in C_P$ according to (7), and m_{I_h} for each set of connected cycles $I_h \in I$ according to (8), and store them in vectors M_{C_P} and M_I, respectively. These vectors are used by the path following strategy to enforce the deadlock avoidance rules, according to *Algorithm 1.* The latter evaluates whether a deadlock is present on a plant state $X(k)$, returning a Boolean $b = 1$.

Algorithm 1. **Deadlock check**
Input: $X(k)$, C_P, D, I, M_{C_P}, M_I. **Output**: $b \in \{0, 1\}$.

1) Initialize $b = 0$; vector $M_{pr} = \{0\}^{|C_P|}$, where each element $m_{pr_j} \in M_{pr}$ is a counter of the parts assuming a state in cycle $c_j \in C_P$ and $|C_P|$ indicates the cardinality of set C_P; vector $M_{imp} = \{0\}^{|I|}$, where each element $m_{imp_h} \in M_{imp}$ is a counter of the parts assuming a state in the set of cycles $I_h \in I$.
2) **If** one or more parts are located in a machine $m \in \mathcal{M}_u$, **then** add fictitious parts states to $X(k)$ corresponding to all the possible paths that can be assigned after the execution of the uncertain job/jobs.
3) Count the number of parts in each cycle $c_j \in C_P$.
For each $c_j \in C_P$, **for** each $n_a^i.n_b^i \in c_j$, **do**:
if one part state in $X(k)$ prescribes transition $n_a^i.n_b^i$, **then** $m_{pr_j} = m_{pr_j} + 1$.
4) **If** $\exists\, m_{pr_j} : m_{pr_j} > m_{c_j}$,
then a deadlock exists, $b = 1$, go to 7).
5) Count the number of parts in each set of cycles $I_h \in I$.
For each $I_h \in I$, **for** each $c_j \in I_h$, **do**:
$m_{imp_h} = m_{imp_h} + m_{pr_j}$.
6) Check whether the cycles in I_h are connected.
For each $m_{imp_h} : m_{imp_h} > m_{I_h}$, **do**:
   6a) **For** each pair of cycles $c_j, c_\ell \in I_h$, $j \neq \ell$, **do**:
   if c_j and c_ℓ have one common node $d_j \cap d_\ell = o$, **then** check whether there exists at least one part involved in the cycles such that the propagation of its state prescribes a transition $n_a.o \in c_j$ immediately followed by $o.n_b \in c_\ell$ before prescribing a transition not involved in the set of connected cycles, and vice versa.
   6b) **If** all the cycles $c_j \in I_h$ are connected,
   then a deadlock exists, $b = 1$, go to 7).
7) Return b.

3.3 Path following strategy

Starting from an initial state $X(k_0)$, at each time step k the path following strategy propagates forward the state $X(k)$ to obtain $X(k+1)$, according to the routing rules, and computes the suitable control inputs $U(k)$. The main idea of the strategy is to check whether any undesired state

occurs after the movement of one part at a time. If it is the case, the part is not allowed to move, meaning that its state is kept the same at the next time step, otherwise it can move and its state is propagated one step forward. The complete strategy is described in *Algorithm 2*.

Algorithm 2. **Path following**
Input: $X(k)$, $a(k)$, $r_m, \forall m \in \mathcal{M}$. **Output**: $U(k)$.

1) Check whether one or more parts have just finished to undergo a job with uncertain outcome.
   For each $i = 1, \dots, N_p(k)$, **for** each $m \in \mathcal{M}_u$, **do**:
   if $\mathcal{S}(s_i(k))^{(1,p_i(k))} = m \wedge r_m = 1$,
   then update the state $x_i(k)$ by assigning to part i a new sequence, and the correct position along it.
2) Initialize the state at next time step: $X(k+1) = X(k)$, and a vector $V = \{0,1\}^{N_p(k)}$. Each $v_i \in V$ indicates whether an action has been assigned to part i. Initially $v_i = 1$ only for parts currently processed in a machine.
3) **While** an action has not been assigned to every part, $\sum_{i=1,\dots,N_p(k)} v_i \neq N_p(k)$, **do**:
   for each part $i = 1, \dots, N_p(k) : v_i = 0$, **do**:
   compute the one-step-ahead prediction of the plant state $\hat{X}$, by forward-propagation of the part state: $\hat{x}_i = [\hat{s}_i, \hat{p}_i]^T = [s_i(k+1), p_i(k+1)+1]^T$ in $X(k+1)$, and perform a series of checks on $\hat{X}$:
   3a) Conflict check.
   if $\exists j,\ j \neq i, : \mathcal{S}(\hat{s}_i)^{(1,\hat{p}_i)} = \mathcal{S}(\hat{s}_j)^{(1,\hat{p}_j)}$,
   then set v_i such that:
   $$v_i = \begin{cases} 1 \text{ if part } j \text{ has an action, } v_j = 1 \\ 0 \text{ otherwise} \end{cases}$$
   and evaluate motion of the next part,
   else go to 3b).
   3b) Deadlock check.
   If the output of *Algorithm 1* on state $\hat{X}$ equals 1, **then** part i holds its position, set $v_i = 1$,
   else part i can move: $X(k+1) = \hat{X}$ and set $v_i = 1$. Evaluate motion of the next part.
4) Compute $U(k)$ for the plant to assume state $X(k+1)$:
   4a) **For** each $n_a, n_b : \exists u_{(n_a,n_b)}$, **do**:
   if $\exists i : \mathcal{S}(s_i(k))^{(1,p_i(k))} = n_a \wedge$
   $\mathcal{S}(s_i(k+1))^{(1,p_i(k+1))} = n_b$,
   then $u_{(n_a,n_b)} = 1$.
   4b) For each input/output station $m \in \mathcal{M}_{I/O}$, check whether one part is ready to enter/exit the plant:
   If $a(k) = 1 \wedge \nexists i : \mathcal{S}(s_i(k+1))^{(1,p_i(k+1))} = m$ and a deadlock is not present on $\hat{X} = [X(k+1), x_{new}]^T$, where x_{new} is the state of the new part,
   then $u_{(0,m)} = 1$ and $X(k+1) = \hat{X}$.
   If $\exists i : \mathcal{S}(s_i(k+1))^{(1,p_i(k+1))} = m \wedge$
   $\mathcal{S}(s_i(k+1))^{(2,p_i(k+1))} = m \wedge r_m = 1$,
   then $u_{(m,0)} = 1$ and remove x_i from $X(k+1)$.
5) Return $U(k)$.

3.4 Fault recovery procedure

In case a fault affecting one transition occurs at time step k, then the Boolean $d(k)$ is set to 1, indicating a fault detection. Subsequently, if alternative sequences to the ones involved in the failure exist, the fault recovery procedure described in *Procedure 1* is run, otherwise the plant operations are interrupted.

Procedure 1. **Fault recovery procedure**

1. Remove the sequences involved in the failure from the set S and replace them with new feasible sequences connecting the same machines.
2. Reassign a suitable state to those parts whose sequence has been replaced. For each of these parts i, if n is the currently occupied node and t the target machine, the new state $x_i(k) = [s_i(k), p_i(k)]^T$ should be such that:
   $$\mathcal{S}(s_i(k))^{(1,p_i(k))} = n \wedge \mathcal{S}(s_i(k))^{(2,p_i(k))} = t.$$
   If a part is on a node that is not part of any sequence, generate a temporary one that takes it from the current node to its target machine. As soon as the part will reach its target, it will be assigned a sequence among the set S.
3. Compute the routing rules for the new set of sequences, also considering the temporary ones.
4. Check whether the reconfigured plant state implies a deadlock or not. If a deadlock is detected, the recovery procedure is stopped, otherwise the operations in the plant can continue normally.

4. REAL CASE IMPLEMENTATION

We consider the automated de-manufacturing plant in the laboratory of the Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing (STIIMA), National Research Council (CNR), in Milano. The plant is designed to test, repair and discharge electronic boards, and comprises four machines: the Load/Unload machine M1, the Testing machine M2, the Repairing machine M3 and the Discharging machine M4. The job scheduling depends on the outcome of the Testing machine, and is described in detail in Boffadossi et al. (2021), as well as the plant-specific handling constraints.

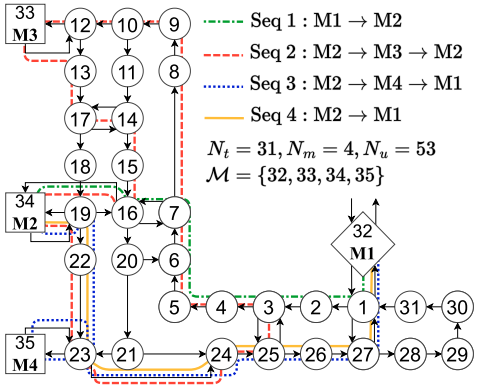


Fig. 5. Sequences of test case on the Plant Graph representing the de-manufacturing plant.

4.1 Test case

Several tests have been executed, with maximum number of parts $N_{p_{max}}$ ranging from 6 to 10. The considered paths are illustrated in Fig. 5, and the machines working times L_m, $m \in \mathcal{M}$ are known: $L_1 = 1$, $L_2 = 5$, $L_3 = 4$ and $L_4 =$

3. The same sequence of outcomes of the Testing machine appears in all the tests, to make the results comparable in terms of throughput. Initially, one part occupies node 32, and the simulations last 2000 time steps. The Boolean signal is always $a(k) = 1$. Finally, the simulations have been performed via MATLAB, using a laptop with 16 GB RAM and an Intel Core i5-1130G7 at 1.8 GHz.

Table 1 reports the results in terms of: the average and the maximum computational time, $C_m\,[s]$ and $C_{peak}\,[s]$; the number of processed parts, N_f; the average and the maximum throughput (parts/time step), T_m and T_{peak}; the number of activated transitions, U_{tot}.

Table 1. Results of test case

$N_{p_{max}}$	C_m	C_{peak}	N_f	T_m	T_{peak}	U_{tot}
6	0.0022	0.0518	203	0.0927	0.1030	6082
7	0.0025	0.0546	204	0.0931	0.1032	6099
8	0.0024	0.0943	203	0.0929	0.1032	6093
9	0.0022	0.0349	204	0.0925	0.1032	6068
10	0.0025	0.0637	201	0.0918	0.1031	6056

The control strategy requires low computational time and the number of machined units N_f is satisfactory. The results also suggest that a higher $N_{p_{max}}$ can decrease the throughput, due to the higher risk of congestion.

The results whit $N_{p_{max}} = 10$ can be compared to those reported in Boffadossi et al. (2021), where a HPRC approach is applied on the plant with the same paths. Specifically, in Boffadossi et al. (2021), when the most aggressive search direction is adopted and the outcomes of the testing machine are deterministic, N_f is slightly higher with respect to this test. However, when the outcomes are uncertain, the same N_f is obtained, but with larger average and peak computational time. These results show that the proposed control strategy achieves good performance even if the optimization level is related to the sequences selection.

4.2 Application of fault recovery procedure

The fault recovery procedure is applied to the considered plant, with sequences shown in Fig. 6 and $N_{p_{max}} = 7$.

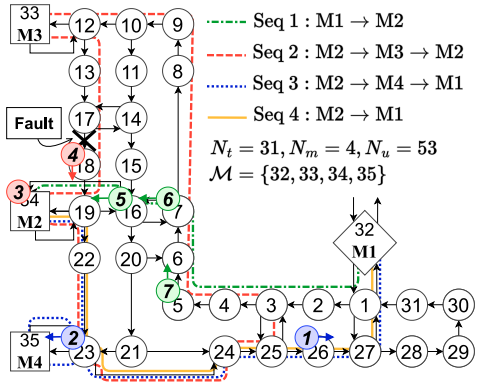


Fig. 6. Initial sequences of fault recovery procedure test. The fault occurs on transition $u_{(17,18)}$.

At time step k the parts are positioned as shown in Fig. 6, when a fault occurs on transition $u_{(17,18)}$, whose activation is prescribed by sequence 2. After its replacement, the resulting sequences correspond to the ones in Fig. 5. Subsequently, the plant state is reconfigured, with the need of adding a temporary sequence from node 18 to 34 for part 4, and new routing rules are computed. The obtained performance is reported in Table 2.

Table 2. Results of fault recovery test

$N_{p_{max}}$	C_m	C_{peak}	N_f	T_m	T_{peak}	U_{tot}
7	0.0032	0.4000	204	0.0931	0.1032	6092

In conclusion, the recovery procedure has been successfully applied, guaranteeing the continuity of the operations without a performance reduction.

5. CONCLUSIONS

A new approach has been proposed for the problem of fault tolerant routing control in a discrete manufacturing plant. The tests performed have reported satisfactory results in terms of throughput and computational efficiency.

Several improvements can be performed in the future, pertaining the introduction of a higher optimization level allocating different sequences to the parts, or the extension of the fault recovery procedure to the cases where deadlocks arise in the reconfigured system.

REFERENCES

Ali, M. and Wadhwa, S. (2010). The effect of routing flexibility on a flexible system of integrated manufacturing. *International Journal of Production Research*, 48(19), 5691–5709.

Boffadossi, R., Fagiano, L., Tanaskovic, M., Cataldo, A., Tanaskovic, M., and Lauricella, M. (2021). Advanced hierarchical predictive routing control of a smart de-manufacturing plant. *2021 European Control Conference (ECC)*, 1774–1779.

Cataldo, A., Morescalchi, M., and Scattolini, R. (2019). Fault tolerant model predictive control of a de-manufacturing plant. *The International Journal of Advanced Manufacturing Technology*, 9(12), 4803–4812.

Cataldo, A. and Scattolini, R. (2016). Dynamic pallet routing in a manufacturing transport line with model predictive control. *IEEE Transactions on Control Systems Technology*, 24(5), 1812–1819.

Fagiano, L., Tanaskovic, M., Mallitasig, L.C., Cataldo, A., and Scattolini, R. (2020). Hierarchical routing control in discrete manufacturing plants via model predictive path allocation and greedy path following. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 5546–5551.

Li, H.X. and Si, H. (2017). Control for intelligent manufacturing: A multiscale challenge. *Engineering*, 3(5), 608–615.

Meindl, B., Ayala, N.F., Mendonça, J., and Frank, A.G. (2021). The four smarts of industry 4.0: Evolution of ten years of research and future perspectives. *Technological Forecasting and Social Change*, 168, 120784.

Morel, G., Valckenaers, P., Faure, J.M., Pereira, C.E., and Diedrich, C. (2007). Manufacturing plant control challenges and issues. *Control Engineering Practice*, 15(11), 1321–1331.

Silva, D.B., Vieira, A.D., Loures, E.F., Busetti, M.A., and Santos, E.A. (2011). Dealing with routing in an automated manufacturing cell: a supervisory control theory application. *International Journal of Production Research*, 49(16), 4979–4998.

Xiong, H., Zhou, M., and Caudill, R. (1996). A hybrid heuristic search algorithm for scheduling flexible manufacturing systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, 2793–2797 vol.3.

Yadav, A. and Jayswal, S. (2018). Modelling of flexible manufacturing system: a review. *International Journal of Production Research*, 56(7), 2464–2487.

Zhang, W., Judd, R.P., and Deering, P.E. (2006). Evaluating order of circuits for deadlock avoidance in a flexible manufacturing system. *International Journal of Production Research*, 44(24), 5247–5259.