Multi-robot routing and scheduling with temporal logic and synchronization constraints

1st Alessio Mosca
University of Pavia
Pavia, Italy
alessio.mosca01@ateneopv.it

2nd Cristian-Ioan Vasile
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
cvasile@mit.edu

3rd Calin Belta
Boston University
Boston, MA 02215, USA
cbelta@bu.edu

4th Davide M. Raimondo
University of Pavia
Pavia, Italy
davide.raimondo@unipv.it

***Abstract*—We consider the problem of scheduling and planning the motion of a fleet of robots involved in semiconductor manufacturing. The robots are tasked with transportation demands of goods that must be fulfilled according to given time constraints and synchronization rules. The demands and synchronization rules are specified using Time Window Temporal Logic (TWTL). Inspiring by model checking techniques, we develop a solution able to guarantee to satisfy both demands and synchronization rules. The proposed approach is tested through simulation on a semiconductor production site of the European project "Power Semiconductor and Electronics Manufacturing 4.0 - (SemI40)" .**

***Index Terms*—Time Window Temporal Logic, Routing and Scheduling, Synchronization, Manufacturing**

I. Introduction

The manufacturing industry has become increasingly competitive over the last decades. In order to survive in such an environment, manufactures must satisfy customer demands exactly in time, quality and quantity. Generally, the production of goods is a complex task which requires several steps. In a medium-complexity semiconductor fab, the process needs 250-500 steps and uses from 50 to 120 different machines [1]. Furthermore, in order to maximize plant efficiency, several goods are usually processed/transported at the same time. In this challenging scenario, the optimal routing and scheduling of goods are of major importance. While in the past the dispatching was performed mainly by human operators, the advance of technology is allowing to move towards fully automated transportation (e.g. fleet of robots). In this case, the routing and scheduling problem consists in finding the optimal robot routes able to minimize transport time while fulfilling constraints (i.e. deadlines, priorities, etc.).

Recently, there has been an increasing interest in using approaches based on temporal logics in motion planning and robot control applications [2, 3, 4]. Temporal logics provide formal high-level languages to describe complex problems, such as the routing and scheduling problems. When explicit time constraints are involved, e.g. the pick up and delivery of a product need to be completed within 10 minutes or before time T=10 min, temporal logics with explicit time must be adopted. Examples include Metric Temporal Logic (MTL) [5], the Signal Temporal Logic (STL) [6] and Time Window Temporal Logic (TWTL) [2].

In this work we use a temporal logic with explicit time to solve routing and scheduling problems [7, 8] for the case where some demands must be delivered in a synchronous way by the robots. We capture the motions of the robots by using weighted Transition Systems (TS)[9]. The transport demands within the production site are expressed by using TWTL. This choice is motivated by the TWTL capability to deal with both time constraints and possible unsatisfiable tasks. In fact, it may unfortunately occur that some demands can not be satisfied due to possible strict time constraints. TWTL provides *temporal relaxation*, which offers the possibility of finding the minimally relaxed formulae (in terms of time constraints) that the robots can satisfy. The TWTL formulae are translated to finite state automata which are then composed with the TS in order to obtain Product Automata (PA)[9]. On the resulting PA the optimal routing and scheduling solution is found by using the Dijkstra's algorithm and selecting among the accepting states of PA the one that can be reached in the shortest time, thus minimizing time constraints relaxation.

This work is related to [4, 10], where the authors rely on Mixed Integer Linear Programming to obtain the vehicle trajectories satisfying the transport demands. In particular, the demands are expressed by using MTL. However, the authors do not consider neither the possibility to find a solution when the demands can not be satisfied due to strict time constraints, nor the capability to guarantee the synchronized delivery. The problem of least-violating planning using linear programming is considered in [11, 12] but their proposed approach does not take into account the synchronization demands. In [2, 13] TWTL is defined and then used for solving the persistent vehicle routing problem but the authors do not take into account the possibility to have the synchronization demands.The closest related work is [14], where the problem of plan synthesis for multi-agent systems in presence of synchronization demand is solved by using Linear Temporal Logic (LTL). However, this approach considers the assignment of the tasks among the robots as prior knowledge and does not allow to deal with explicit time constraints.

II. Preliminaries

For a finite set Σ, we denote 2^{Σ} and $|\Sigma|$ the set of all subsets, and the cardinality of Σ, respectively. A *word* $\boldsymbol{\sigma}$ is a finite or infinite sequence of elements from Σ. Let $|\boldsymbol{\sigma}|$ indicate

the length of a word $\boldsymbol{\sigma}$. The repetition of symbol σ d times is denoted by $\sigma^{\{d\}}$. A *language* is a set of words over the *alphabet* Σ.

Definition 2.1 (Deterministic Transition System, DTS): A (weighted) Deterministic Transition System is a tuple $\mathcal{T} = (Q, q_0, \Delta, AP, h, \omega)$, where Q is the finite set of states; $q_0 \in Q$ is the initial state; $\Delta \subseteq Q \times Q$ is the set of transitions; AP is the set of observations (atomic propositions); $h : Q \to 2^{AP}$ is the labeling function and; $\omega : \Delta \to \mathbb{Z}_{>0}$ is a map that assigns a positive integer weight to each transition.

A transition $(q, q') \in \Delta$ of $\mathcal{T}$ is also denoted by $q \to_{\mathcal{T}} q'$. We define a $trajectory$ of the system as a sequence of states $\mathbf{q} = q_0, q_1, \ldots$ such that $q_k \to_{\mathcal{T}} q_{k+1}$ for all $k \geq 0$. A $trajectory$ generates an *output word* $\mathbf{o} = \mathbf{o}_0 \cdot \mathbf{o}_1 \cdot \mathbf{o}_2 \ldots$, where $\mathbf{o}_0 = h(q_0)$, $\mathbf{o}_k = h(q_k)^{\{\omega((q_k, q_k))\}}$ if $q_k = q_{k-1}$, and $\mathbf{o}_k = \emptyset^{\{\omega((q_{k-1}, q_k))-1\}} h(q_k)$ if $q_k \neq q_{k-1}$ for all $k \geq 1$. The sub-word $\mathbf{o}_k$ corresponds to the observations generated along the transition q_{k-1}, q_k of duration $\omega((q_{k-1}, q_k))$. Note that, as opposed to state trajectories, output words are defined at each discrete time $k \in \mathbb{Z}_{\geq 0}$, where the weights of $\mathcal{T}$ are interpreted as transition durations. Thus, we consider no observations (i.e., $\emptyset$) along transitions. We also denote $\mathbf{o}$ by $h(\mathbf{q})$. Let $\mathcal{L}(\mathcal{T})$ be the set of all output words generated by $\mathcal{T}$, i.e., its *generated language*. Let $\mathbf{q}_1, \ldots, \mathbf{q}_m$ be trajectories of $\mathcal{T}_1, \ldots, \mathcal{T}_m$ with the same alphabet AP, and $\mathbf{o}_\ell = h_\ell(\mathbf{q}_\ell) = o_{\ell,0}, o_{\ell,1} \ldots$ the corresponding output words for all $\ell \in \{1, \ldots, m\}$. The joint output word generated by all trajectories is $h(\times_{\ell=1}^{m} \mathbf{q}_\ell) = o_0, o_1, \ldots$, where $o_k = (\bigcup_{\ell=1}^{m} o_{\ell,k}) \in 2^{AP}$, and $k \in \{0, 1, \ldots\}$ indicates the time instants at which observations occur.

Definition 2.2 (Time Window Temporal Logic): A Time Window Temporal Logic formula over a set of atomic propositions AP is defined as follows

$$\varphi ::= \mathbf{H}^d s \,|\, \mathbf{H}^d \neg s \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \vee \varphi_2 \,|\, \varphi_1 \cdot \varphi_2 \,|\, [\varphi_1]^{[a,b]},$$

where $s \in AP \cup \{\top\}$ is either an atomic proposition or the "true" constant $\top$; $\neg, \wedge, \vee$ are the negation, conjunction, and disjunction Boolean operators, respectively; $\cdot$ is the concatenation operator; $[\varphi_1]^{[a,b]}$ with $0 \leq a \leq b$ is the *within* operator, and $\mathbf{H}^d$ with $d \geq 0$ is the *hold* operator. When $d = 0$, we drop $\mathbf{H}$ from the notation, e.g., $s \equiv \mathbf{H}^0 s$. The satisfaction of an TWTL formula φ is defined with respect to finite output words $\mathbf{o}$ over 2^{AP}. The hold operator $\mathbf{H}^d s$ is satisfied if $s \in AP$ is repeated for d time units. Instead, the $\mathbf{H}^d \neg s$ requires that for d time units only symbols from $AP \setminus \{s\}$ appear. The within operator $[\varphi]^{[a,b]}$ is satisfied if the formula φ becomes true in the given time window $[a, b]$. The concatenation operator $\varphi_1 \cdot \varphi_2$ requires that formula φ_1 is first satisfied and then φ_2 is satisfied immediately after. The Boolean operators have their usual semantics.

A complete description of the semantics of TWTL can be found in [15].

The satisfaction of a TWTL formula can be decided in bounded time. We denote the maximum time needed to satisfy ϕ by $\|\phi\|$, which can be computed as follows:

$$\|\phi\| = \begin{cases} d & \text{if } \phi \in \{\mathbf{H}^d s, \mathbf{H}^d \neg s\} \\ \max(\|\phi_1\|, \|\phi_2\|) & \text{if } \phi \in \{\phi_1 \wedge \phi_2, \phi_1 \vee \phi_2\} \\ \|\phi_1\| & \text{if } \phi = \neg \phi_1 \\ \|\phi_1\| + \|\phi_2\| + 1 & \text{if } \phi = \phi_1 \cdot \phi_2 \\ b & \text{if } \phi = [\phi_1]^{[a,b]} \end{cases} \quad (1)$$

Let $\phi = \left[M_1 \cdot [M_2]^{[c,d]}\right]^{[a,b]}$ be a TWTL formula that describes, for example, a possible transport demand from the pickup point M_1 to the delivery point M_2. Note that every time there is a concatenation between two formulas, the time constraints of the second formula are relative to the time when the first one was satisfied. In the previous example one has "satisfy M_2 between c and d time instants after the satisfaction of M_1". The formula will be satisfied if and only if all the sub-tasks (pickup and delivery) are fulfilled within the time window expressed by the external *within* operator, i.e. [a,b]. What if ϕ can not be fulfilled in the given time window? In order to cope with this possible problem, in [15] the authors propose *temporal relaxation* of TWTL formulae. The *temporal relaxation* introduces the possibility to relax the deadlines for the time windows, which are expressed by the *within* operator. Thus, the relaxed version of ϕ is $\phi(\boldsymbol{\tau}) = \left[M_1 \cdot [M_2]^{[c,d+\tau_2]}\right]^{[a,b+\tau_1]}$, where $\boldsymbol{\tau} = (\tau_1, \tau_2) \in \mathbb{Z}^2$. Furthermore, the $\phi(\boldsymbol{\tau})$ has to preserve the feasibility of ϕ, i.e. every time window of a within operator has to be greater or equal than the time needed to satisfy the task enclosed by the within operator. Note that, in the following we allow only the relaxation of the deadlines (upper bound) because the relaxation of the lower bound of a within operator would correspond to anticipating the pickup time of an order. Unfortunately, this is not possible in general since the order may not be already available.

Definition 2.3 (τ-relaxation of ϕ): Given a TWTL formula ϕ with m *within* operators, the feasible τ-relaxation of ϕ is defined as $\phi(\boldsymbol{\tau})$, where $\boldsymbol{\tau} \in \mathbb{Z}^m$ and each subformula of the form $[\phi_i]^{[a_i,b_i]}$ is replaced with $[\phi_i]^{[a_i,b_i+\tau_i]}$ for all $i \in \{1, \ldots, m\}$.

Definition 2.4 (Linear Temporal Relaxation): Given ϕ, let $\phi(\boldsymbol{\tau})$ be the feasible relaxation of ϕ. The linear temporal relaxation of ϕ is $|\boldsymbol{\tau}|_{LTR} = \sum_{i=1}^{m} \tau_i$.

Definition 2.5 (Deterministic Finite State Automaton): A Deterministic Finite State Automaton (DFA) is a tuple $\mathcal{A} = (S_\mathcal{A}, s_0, \delta_\mathcal{A}, 2^{AP}, F_\mathcal{A})$, where $S_\mathcal{A}$ is a finite set of states; $s_0 \in S_\mathcal{A}$ is the initial state; $\delta_\mathcal{A} : S_\mathcal{A} \times 2^{AP} \to S_\mathcal{A}$ is the transition function; 2^{AP} is the input alphabet; and $F_\mathcal{A} \subseteq S_\mathcal{A}$ is the set of (final) accepting states.

We denote a transition from s to $s' = \delta_\mathcal{A}(s, \sigma)$ with input symbol σ as $s \xrightarrow{\sigma}_\mathcal{A} s'$. A finite sequence of symbols $\boldsymbol{\sigma} = \sigma_0, \sigma_1, \ldots, \sigma_n$ generates a trajectory of the DFA $\mathbf{s} = s_0, s_1, \ldots, s_n$ such that $s_0 \in S_\mathcal{A}$ is the initial state of $\mathcal{A}$, and $s_k \xrightarrow{\sigma}_\mathcal{A} s_{k+1}$ denotes the transition from time k to $k+1$. The word $\boldsymbol{\sigma}$ is accepted by the DFA if and only if the

corresponding trajectory ends in the final automaton state, i.e., $\sigma_{n+1} \in F_{\mathcal{A}}$. The accepted language of the DFA $\mathcal{A}$ is defined as $\mathcal{L}(\mathcal{A})$.

Formulae expressed in TWTL can be captured by DFAs as shown in [15]. Methods to compute DFAs accepting possible deadline relaxations, and to perform synthesis and verification using a *bottleneck* temporal relaxation cost have been proposed in [15]. Here, we employ the automata construction methods, but consider a linear temporal relaxation cost instead.

III. Problem Formulation

In this section we define the environment, the robot model, and the transport demands that characterize, for example, the dispatching problem in a semiconductor manufacturing fab.

A. Environment Model

Let $\mathcal{G} = (Q, \Delta, \omega)$ denote a weighted directed connected graph, where Q represents the set of locations of interest (machines location, charging stations, interconnection nodes) labeled with observations from AP as given by map $h : Q \to 2^{AP}$. The edges $\Delta \subseteq Q \times Q$ capture feasible motions between locations with nominal travel times given by $\omega = \Delta \to \mathbb{Z}_{\geq 1}$. Travel times are expressed in terms of a global discrete clock with time step Δt.

B. Robot Model

Consider a team of m robots moving in an environment $\mathcal{G}$. The motion model of each robot $v \in \{1, \ldots, m\}$ is captured by a TS $\mathcal{T}_v = (Q, q_{v,0}, \Delta, AP, h, \omega)$, where $q_{v,0} \in Q$ is the initial state of the v-th robot, and ω is its deterministic travel time function such that $\omega = \Delta \to \mathbb{Z}_{\geq 1}$.

We assume that all robots can communicate with all other robots. Furthermore, each robot is able to detect its position when it reaches a node of interest $q \in Q$. In the following, we assume that each robot can transport (fulfill) at most one transport demand at a time, i.e., robots have single capacity.

C. Specification: Transportation Demands and Synchronization Rules

Let $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n\}$ be the set of the n transport demands that must be satisfied. The i-th demand is defined as the tuple $\mathcal{D}_i = (\phi_i, \pi_i^{start})$, where ϕ_i is a TWTL formula, and $\pi_i^{start} \in AP$ indicates the start proposition of the formula ϕ_i, i.e., the pick-up location. For brevity, we assume that transportation demand formulae include the pick-up specification, i.e., are of the form $\phi_i = [\pi_i^{start} \wedge \phi_i']^{[0, \|\phi_i'\|]}$, where ϕ_i' is a TWTL formula.

Since some elements of $\mathcal{D}$ may require to be fulfilled (i.e. demands delivered) at the same time, we define a set $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots\}$ of synchronization rules. The j-th rule is defined as tuple $\mathcal{R}_j = (\psi_j, I_j)$, where ψ_j is the TWTL formula of the task to be performed in a synchronous way, and $I_j \subseteq \{1, \ldots, n\}$ indicates which elements of $\mathcal{D}$ are involved in ψ_j. Let $\widehat{\psi}_j = \left(\bigwedge_{\ell \in I_j} [\pi_\ell^{start}]^{[0, \|\phi_\ell\|]}\right) \cdot \psi_j$ with the meaning that the synchronization task ψ_j of rule $\mathcal{R}_j$ must be satisfied after the start of all associated transportation demands in I_j.

The overall specification, in which all transport demands and synchronization rules are considered, is expressed as

$$\varphi = \Phi \wedge \Psi, \tag{2}$$

where $\Phi = \bigwedge_{i=1}^{|\mathcal{D}|} \phi_i$, and $\Psi = \bigwedge_{j=1}^{|\mathcal{R}|} \widehat{\psi}_j$.

D. Problem Definition

Given a set of demands to be satisfied and a list of synchronization rules, our goal is to find the optimal assignment of demands and the corresponding optimal paths for the different robots. Optimality is with respect to the total deadline deviations over all demands and rules.

Problem 3.1: Given φ, i.e. the specification of transportation demands and synchronization rules as in (2), an environment $\mathcal{G}$, m robots modelled as $\mathcal{T}_1, \ldots, \mathcal{T}_m$, find trajectories $\text{Traj} = \{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ such that φ is satisfied with minimum temporal relaxation

$$\begin{aligned} \min_{\text{Traj}} \quad & |\boldsymbol{\tau}|_{LTR} \\ \text{subject to} \quad & \forall \mathcal{D}_i \in \mathcal{D}, \exists \mathbf{q}_{r_i} \in \text{Traj} \ : \ h(\mathbf{q}_{r_i}) \models \phi_i(\boldsymbol{\tau}_{\phi_i}) \\ & \forall \mathcal{R}_j \in \mathcal{R} \ : \ h(\times_{\ell \in I_j} \mathbf{q}_{r_\ell}) \models \widehat{\psi}_j(\boldsymbol{\tau}_{\psi_j}), \end{aligned}$$

where $\boldsymbol{\tau} = [\boldsymbol{\tau}_{\phi_1}, \ldots, \boldsymbol{\tau}_{\phi_{|\mathcal{D}|}}, \boldsymbol{\tau}_{\psi_1}, \ldots, \boldsymbol{\tau}_{\psi_{|\mathcal{R}|}}]$ is the vector of all deadline relaxation variables for transportation demands and synchronization rules.

IV. Solution

In this section, we present the solution of Problem 3.1 First, for each robot v we create the deterministic motion transition system $\mathcal{T}_v$. In particular, we define this latter so to have all edges with weight one. This is achieved by replacing original transitions $e \in \Delta$ with $\omega(e)$ transitions. Second, the transportation demands and synchronization rules are translated to DFAs. The i-th automaton obtained from ϕ_i is denoted by $\mathcal{A}_{\phi_i}$, while $\mathcal{A}_{\widehat{\psi}_j}$ refers to the j-th automaton of Ψ obtained from $\widehat{\psi}_j = \left(\bigwedge_{\ell \in I_j} [\pi_\ell^{start}]^{[0, \|\phi_\ell\|]}\right) \cdot \psi_j$. Next, we construct a product automaton $\mathcal{P}$ as defined in Definition 4.1 that captures the motion of the m robots and the satisfaction of the specification φ from (2). On the resulting product automaton, the solution in terms of optimal assignments and shortest paths is then found using Dijkstra's algorithm.

Definition 4.1 (Product Automaton): Given the product transition system $\mathcal{T}^m = \bigtimes_{v=1}^{m} \mathcal{T}_v = (Q^m, q_0^m, \Delta^m, 2^{AP}, h^m)$, the automata $\mathcal{A}_{\phi_i} = (S_{\mathcal{A}_{\phi_i}}, s_{0,i}, \delta_{\mathcal{A}_{\phi_i}}, 2^{AP}, F_{\mathcal{A}_{\phi_i}})$, for all $i = 1, \ldots, |\mathcal{D}|$, and the automata $\mathcal{A}_{\widehat{\psi}_j} = (S_{\mathcal{A}_{\widehat{\psi}_j}}, \widehat{s}_{0,j}, \delta_{\mathcal{A}_{\widehat{\psi}_j}}, 2^{AP}, F_{\mathcal{A}_{\widehat{\psi}_j}})$, for all $j = 1, \cdots, |\mathcal{R}|$, the product automaton (PA) is a tuple $\mathcal{P} = (S_{\mathcal{P}}, s_{0,\mathcal{P}}, \Delta_{\mathcal{P}}, F_{\mathcal{P}}, \omega_{\mathcal{P}})$, where

- $S_{\mathcal{P}} = Q^m \times \left(S_{\mathcal{A}_{\phi_i}} \times \{0, \cdots, m\}\right)_{i=1}^{|D|} \times \left(S_{\mathcal{A}_{\widehat{\psi}_j}}\right)_{j=1}^{|R|}$ is the finite set of states;
- $s_{0,\mathcal{P}} = \left(x_{-1}, (s_{0,i}, 0)_{i=1}^{|D|}, (\widehat{s}_{0,j})_{j=1}^{|R|}\right)$ is the initial state;
- $\Delta_{\mathcal{P}} \subseteq S_{\mathcal{P}} \times S_{\mathcal{P}}$ is a transition relation. Let r_i be the robot assigned to i-th demand. Then $\left(x, (s_i, r_i)_{i=1}^{|D|}, (\widehat{s}_j)_{j=1}^{|R|}\right) \to_{\mathcal{P}} \left(x', (s_i', r_i')_{i=1}^{|D|}, (\widehat{s}_j')_{j=1}^{|R|}\right) \in \Delta_{\mathcal{P}}$ iff:

- $x = (q_1, \ldots, q_m)$, $x' = (q'_1, \ldots, q'_m)$, $q_v \rightarrow_{\mathcal{T}_v} q'_v \in \Delta$, $\forall v \in \{1, \ldots, m\}$;
- $(r_i = 0 \ \wedge \ s_i = s_{0,i} \ \wedge \ s_i \xrightarrow{\sigma_i} s'_i \ \wedge \ \sigma_i = h(q'_v) = \pi_i^{start} \wedge \ r'_i = v) \ \vee$
  $(r_i = v \ \wedge \ s_i \xrightarrow{\sigma_i} s'_i \ \wedge \ \sigma_i = h(q'_v) \ \wedge \ r'_i = v)$, $\forall i \in \{1, \ldots, |D|\}$;
- $\widehat{s}_j \xrightarrow{\widehat{\sigma}_j} \widehat{s}'_j \ \wedge \ \widehat{\sigma}_j = \{h(q'_{r_i}) \mid i \in I_j \wedge r_i > 0\}$, $\forall j \in \{1, \cdots, |R|\}$;

- $\omega_{\mathcal{P}}(s_{\mathcal{P}}, s'_{\mathcal{P}}) = |D| - \sum_{i=1}^{|D|} |\{s'_i\} \cap F_{\phi_i}| + c(x, x')$ is the weight function, where $\sum_{i=1}^{|D|} |\{s'_i\} \cap F_{\phi_i}|$ is the number of fulfilled demands in s', and $c(x, x')$ is a cost used to penalize the number of robots changing positions in the transition from x to x' so to avoid unnecessary movements;
- $F_{\mathcal{P}} = Q^m \times (F_{\phi_i} \times \{1, \ldots, m\})_{i=1}^{|D|} \times \left(F_{\widehat{\psi}_j}\right)_{j=1}^{|R|}$ is the set of accepting states.

For simplicity, we introduce the initial state $x_{-1} = (q_{1,-1}, \ldots, q_{m,-1}) \in Q^m$ such that, for all $v \in \{1, \ldots, m\}$, the only transition available from $q_{v,-1}$ is $q_{v,-1} \rightarrow_{\mathcal{T}_v} q_{v,0}$.

Similar to TS, a trajectory of $\mathcal{P}$ is a sequence $\mathbf{p} = p_0, p_1, \ldots$ such that $p_0 = s_{0,\mathcal{P}}$ and $(p_k, p_{k+1}) \in \Delta_{\mathcal{P}}$ for all $k \geq 0$. Any satisfying (accepted) trajectory of $\mathcal{P}$ ends in a state of $F_{\mathcal{P}}$. The solution of Problem 3.1 is obtained by computing an optimal satisfying trajectory $\mathbf{p}^*$ using the Dijkstra's algorithm and selecting among the accepting states of PA the one that can be reached in the shortest time, thus minimizing time constraints relaxation. By construction, $\mathbf{p}^*$ encodes valid movements of robots in the environment $\mathcal{G}$, i.e., transitions in Δ, and satisfies all transportation demands and synchronization rules. The trajectories that robots have to follow are obtained by projecting $\mathbf{p}^*$ onto each $\mathcal{T}_v$, $v \in \{1, \ldots, m\}$, as given by Definition 4.2.

Definition 4.2: (Projection of a trajectory of $\mathcal{P}$ onto $\mathcal{T}_v$). Let $\mathbf{p} = p_0, p_1, \ldots$ be a trajectory of $\mathcal{P}$, where $p_k = \left(x_k, (s_{i,k}, r_{i,k})_{i=1}^{|D|}, (\widehat{s}_{j,k})_{j=1}^{|R|}\right)$ and $x_k = (q_{1,k}, \ldots, q_{m,k})$. The projection of $\mathbf{p}$ onto $\mathcal{T}_v$ is the trajectory $\mathbf{q}_v = q_{v,0}, q_{v,1}, \ldots$ for all $v \in \{1, \ldots, m\}$.

Algorithm 1 summarizes the solution to Problem 1.

Algorithm 1: Solution

Input : $\mathcal{G}$ - the environment
Input : $\mathcal{D}$ - the set of demand to be satisfied
Input : $\mathcal{R}$ - the set of synchronization rules
Output: The optimal run $\mathbf{q}_v$ for each robot $v \in \{1, \ldots, m\}$

1 Construct the TSs $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ for all robots
2 Construct the FSAs $\{\mathcal{A}_{\phi_1}, \cdots, \mathcal{A}_{\phi_{|D|}}\}$ corresponding to $\mathcal{D}$
3 Construct the FSAs $\{\mathcal{A}_{\psi_1}, \cdots, \mathcal{A}_{\psi_{|R|}}\}$ corresponding to $\mathcal{R}$
4 Construct the product automaton $\mathcal{P}$ as defined in Definition 4.1
5 Find the shortest trajectories from the initial condition $s_{0,\mathcal{P}}$ to the accepting states $F_{\mathcal{P}}$ of PA using Dijkstra's algorithm and select the optimal one $\mathbf{p}^*$ in terms of time relaxations
6 Project the optimal trajectory $\mathbf{p}^*$ onto $\mathcal{T}_1, \ldots, \mathcal{T}_m$ as defined in Definition 4.2

Remark. The collision avoidance can be achieved by removing unwanted states and transitions of $\mathcal{P}$ (Definition 4.1). For simplicity, we do not consider collision avoidance in this paper, and leave it for future work.

A. Complexity

In Algorithm 1 the computational complexity for constructing the vehicle transition system $\mathcal{T}$ is $\mathcal{O}\left(\sum_e \omega(e)\right)$, where ω is the travel times. The size of the transition system $\mathcal{T}^m$ that models m identical vehicles is $\mathcal{O}\left(|Q|^m\right)$. The time complexity for translating a formula ϕ into FSA is $\mathcal{O}(2^{|\phi|})$, where $|\phi|$ is the length of the formula [15]. Finally, the time complexity of $\mathcal{P}$ is $\mathcal{O}\left(|Q|^m \cdot m \cdot |\mathcal{D}| \cdot \left|S_{\mathcal{A}_\phi}\right| \cdot |\mathcal{R}| \cdot \left|S_{\mathcal{A}_\psi}\right|\right)$, while the time complexity of Dijkstras algorithm is $\mathcal{O}(|S_{\mathcal{P}}| \log |S_{\mathcal{P}}|)$.

V. Simulations and Results

The algorithms presented in this work are implemented in Python2.7 using the PyTWTL package [15]. All simulations were performed on a MacBook Pro with i5 @2.09 GHz 64bit CPU, 8 GB of RAM, and MacOS Mojave.

In this section, we present simulation results in a semiconductor production site of SemI40 project, which is the sponsor of this work. The end products of the semiconductor manufacturing process are integrated circuits. Chips are composed of several layers of chemical patterns that are imprinted on silicon wafers by machines. To obtain a layer, it is necessary that the wafer undertakes several steps, e.g., deposition, photolithography, and etching, performed by the machines. Since the cost of the machines is prohibitive, the wafers must revisit the machines multiple times to obtain the end products. Moreover, some steps must be performed at the same time on a machine (e.g. the diffusion furnaces [16]). We employ a fleet of robots to perform transportation demands between machines, and satisfy the synchronization rules of the fabrication process. The objective of these latter is to avoid time losses thus maximizing efficiency in the fabrication process. The challenge in the semiconductor fab is to find the optimal routing and scheduling for meeting the transport demands and the synchronization rules. Robots are responsible for the demand transportation among the various machines present in the fab.

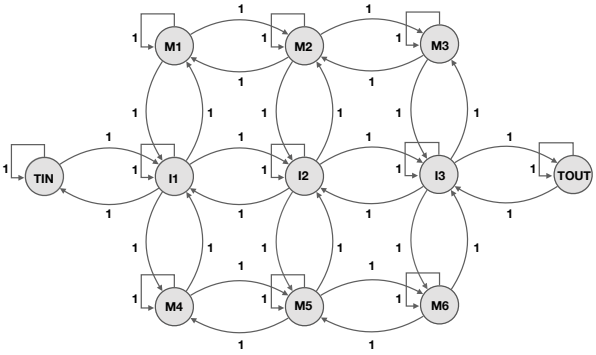

Fig. 1: The environment transition system $\mathcal{T}_v$

TABLE I: Transport Demands ($\mathcal{D}$) and Synchronization Rules ($\mathcal{R}$)

$\mathcal{D}$	$\mathcal{R}$
$\phi_1 = \left[T_{IN} \cdot [M_5]^{[0,4]}\right]^{[0,6]}, \pi_1^{start} = T_{IN}$	$\psi_1 = [M_5]^{[0,6]}, I_1 = \{1,3\}$
$\phi_2 = \left[M_6 \cdot [T_{OUT}]^{[0,5]}\right]^{[0,7]}, \pi_2^{start} = M_6$	$\psi_2 = [T_{OUT} \wedge M_3]^{[0,7]}, I_2 = \{2,5\}$
$\phi_3 = \left[M_4 \cdot [M_5]^{[0,4]}\right]^{[0,6]}, \pi_3^{start} = M_4$	
$\phi_4 = \left[M_2 \cdot [M_4]^{[0,2]}\right]^{[0,4]}, \pi_4^{start} = M_2$	
$\phi_5 = \left[M_5 \cdot [M_3]^{[0,5]}\right]^{[0,7]}, \pi_5^{start} = M_5$	

TABLE II: Optimal Nominal Solution $\mathbf{p}^*$

$\mathcal{T}^2$		ϕ_1		ϕ_2		ϕ_3		ϕ_4		ϕ_5		$\widehat{\psi}_1$	$\widehat{\psi}_2$
$\mathbf{q}_1$	$\mathbf{q}_2$	$\mathcal{A}_{\phi_1}$	r_i	$\mathcal{A}_{\phi_2}$	r_i	$\mathcal{A}_{\phi_3}$	r_i	$\mathcal{A}_{\phi_4}$	r_i	$\mathcal{A}_{\phi_5}$	r_i	$\mathcal{A}_{\widehat{\psi}_1}$	$\mathcal{A}_{\widehat{\psi}_2}$
M_1	T_{OUT}	s_0	0	s_0	0	s_0	0	s_0	0	s_0	0	s_0	s_0
M_1	I_3	s_0	0	s_0	0	s_0	0	s_0	0	s_0	0	s_0	s_0
M_2	I_2	s_0	0	s_0	0	s_0	0	s_1	1	s_0	0	s_0	s_0
I_2	I_1	s_0	0	s_0	0	s_0	0	s_1	1	s_0	0	s_0	s_0
I_2	T_{IN}	s_1	2	s_0	0	s_0	0	s_1	1	s_0	0	s_0	s_0
M_4	I_1	s_1	2	s_0	0	s_0	0	s_F	1	s_0	0	s_0	s_0
M_4	I_2	s_1	2	s_0	0	s_1	1	s_F	1	s_0	0	s_0	s_0
M_5	M_5	s_F	2	s_0	0	s_F	1	s_F	1	s_0	0	s_F	s_0
M_5	M_5	s_F	2	s_0	0	s_F	1	s_F	1	s_1	1	s_F	s_0
I_2	M_6	s_F	2	s_1	2	s_F	1	s_F	1	s_1	1	s_F	s_0
I_3	I_3	s_F	2	s_1	2	s_F	1	s_F	1	s_1	1	s_F	s_0
M_3	T_{OUT}	s_F	2	s_F	2	s_F	1	s_F	1	s_F	1	s_F	s_F

Consider a production site composed of 6 machines $M_1, \ldots, M_6$, 1 Transfer Point In T_{IN}, and 1 Transfer Point Out T_{OUT}. Demands processed in the fab sector appear at T_{IN} and are released at T_{OUT}, respectively. Each demand must follow its specific recipe based on following information available at its arrival: (a) pickup position, (b) delivery position, (c) wafer transportation time window within, and (d) any synchronization requirements with other demands. We abstract the fab sector environment into a weighted graph, where the nodes represent points of interest (machines and transfer points), edges indicate the possibility of motion between nodes, and the weights represent travel times associated with the edges. The motion model of each robot is abstracted as a transition system obtained from the environment graph by splitting each edge into a number of transitions equal to the corresponding edges nominal travel time, see Figure 1. The set of propositions (AP) is $AP = \{M_1, M_2, M_3, M_4, M_5, M_6, T_{IN}, T_{OUT}\}$.

We consider 2 robots that must fulfill 5 transportation demands subject to 2 synchronization rules shown in Table I. All transportation demands and synchronization rules are captured by TWTL formulae, and translated to FSAs.

The optimal trajectory $\mathbf{p}^*$ of $\mathcal{P}$ satisfying the transportation demands and synchronization rules is computed using Algorithm 1 and shown in Table II.

The minimal temporal relaxation vector associated with $\mathbf{p}^*$ is $\boldsymbol{\tau} = \left(\boldsymbol{\tau}_{\phi_1}, \ldots, \boldsymbol{\tau}_{\phi_5}, \boldsymbol{\tau}_{\widehat{\psi}_1}, \boldsymbol{\tau}_{\widehat{\psi}_2}\right) = (1,4,1,1,4,1,4)$ and the minimum linear temporal relaxation is $|\boldsymbol{\tau}|_{LTR} = 16$. Table II shows that the transportation demands ϕ_4, ϕ_3 and ϕ_5 are satisfied by the robot 1, while ϕ_1 and ϕ_2 are satisfied by the robot 2. Table III shows the runtime performance.

TABLE III: Quantitative information on scalability

	Number of states	Number of transitions	Computational time
$\mathcal{T}$	11	39	8 ms
$\mathcal{T}^2$	121	1521	16 ms
$\mathcal{A}_{\phi_i}$	3	4	11 ms
$\mathcal{A}_{\psi_j}$	2	2	16 ms
$\mathcal{P}$	23185	295802	58.4 s

VI. Conclusion

We have studied the problem of planning and scheduling for a fleet of robots operating in a manufacturing systems, where some demands require to be satisfied at the same time. Inspiring by model checking techniques, we have proposed a solution where both demands and synchronization rules are satisfied. Future work will be devoted to develop an approach able to guarantee the satisfaction of the synchronization rules for a fleet of robots with motion uncertainty.

References

[1] F. Thiesse and E. Fleisch, "On the value of location information to lot scheduling in complex manufacturing processes," *International Journal of Production Economics*, vol. 112, no. 2, pp. 532–547, 2008.

[2] C. I. Vasile and C. Belta, "An automata-theoretic approach to the vehicle routing problem." in *Robotics: Science and Systems*, 2014.

[3] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[4] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 3953–3958.

[5] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[6] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[7] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Procs. of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[8] P. Toth and D. Vigo, *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[9] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[10] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

[11] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *International Conference on Hybrid systems: computation and control*. ACM, 2013, pp. 1–10.

[12] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *Procs. of the 7th International Conference on Cyber-Physical Systems*. IEEE, 2016, p. 17.

[13] D. Aksaray, C.-I. Vasile, and C. Belta, "Dynamic routing of energy-aware vehicles with Temporal Logic Constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3141–3146.

[14] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[15] C.-I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[16] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason, and O. Rose, "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations," *Journal of scheduling*, vol. 14, no. 6, 2011.