Quality Metrics to Evaluate Flexible Timeline-based Plans

Alessandro Umbrico[1], Andrea Orlandini[2], Marta Cialdea Mayer[1]

[1] Dipartimento di Ingegneria
Università degli Studi Roma Tre
[2] Istituto di Scienze e Tecnologie della Cognizione
Consiglio Nazionale delle Ricerche, Roma

Abstract. Timeline-based Planning has been successfully applied in several contexts to solve Planning and Scheduling (P&S) problems. A key enabling feature of the timeline-based approach to planning is its capability of dealing with temporal flexibility. Temporal flexibility is an important feature in real world scenarios. Indeed, it can be exploited by an executive system for robust on-line execution of flexible plans in order to absorb possible delays during the execution. In this regard, it is useful to define quality metrics to evaluate the robustness of flexible timeline-based plans. In this paper, a set of quality metrics for flexible timeline-based plans are defined and discussed when applied to a specific timeline-based framework. In fact, we consider the framework EPSL, developed to support the design of P&S applications, that allows the definition of different planners endowed with specific heuristics. Then, an experimental analysis is presented exploiting the planners to solve planning problem instances related to a real-world manufacturing case study. And, finally, an evaluation of planners performance is presented and discussed comparing results also considering robustness of generated plans.

1 Introduction

The Timeline-based approach to planning has been successfully applied in several real world scenarios, especially in space like contexts [1,2,3,4]. Besides these applications, several timeline-based Planning and Scheduling (P&S) systems have been deployed to define domain specific applications, see for example EUROPA [5], IXTET [6], APSI-TRF [7]. Like in classical planning [8], it is important to consider different aspects of the plans generated by timeline-based P&S systems, in order to evaluate their quality. Some temporal metrics have been defined in the literature for temporal networks (often used to represent timeline-based plans) and scheduling [9,10]. In general these metrics characterize the robustness of a schedule by considering the temporal constraints between its activities.

Analogously, the robustness of a timeline-based plan can be evaluated by means of similar measures, which are to be defined without relying on the underlying temporal network (TN). This is particularly important when time flexibility is taken into account. In fact, representing a flexible timeline-based plan as a TN (or a schedule) entails a sort of simplification of the associated plan structure causing a lost of information on the

"dependencies" among its components which can usefully be taken into account. Such information is useful both to generate the plan, as described in [11], and to make a more detailed analysis of the temporal features which are relevant to evaluate the overall plan robustness. For instance, some timeline in the plan can "dominate" the behavior of other timelines, since its temporal deviations are most likely to propagate to the others.

This paper represents a first step towards the characterization of temporal qualities of flexible timeline-based plans. After a brief presentation of the basic concepts underlying flexible timeline-based planning, some of such metrics are introduced. Their concrete application is shown by using planners implemented in the EPSL (*Extensible Planning and Scheduling Library*) framework [12]. EPSL is a domain independent, modular and extensible software environment supporting the development of timeline-based applications. Recently, some improvements concerning the representation and solving capabilities of EPSL have been presented in [11]. Specifically, the framework is enriched with the capability to model and reason on renewable resources and domain independent heuristics supporting the planning process. The general structure of EPSL allows for preserving "past experiences" by providing a set of ready-to-use algorithms, strategies and heuristics that can be combined together. In this way, it is possible to develop and evaluate different solving *configurations* in order to find the one which better addresses the features of the particular planning problem to be solved. This paper resorts to the EPSL framework and its application to a real-world manufacturing case study, in order to evaluate and compare the performances of planners using different heuristics and the robustness of the generated plans.

2 Flexible Timeline-based Planning

This section briefly introduces the main concepts to define flexible timeline-based plans, along the lines of [13]. The timeline-based approach to P&S aims at controlling a complex system by synthesizing temporal behaviors of its features in terms of timelines. A planning domain is modeled as a set of features, represented by *multi-valued state variables*, that must be controlled over time. Causal and temporal constraints specify, for each feature, the set V of the values the variable may assume, the allowed value transitions (by means of a function $T : V \rightarrow 2^V$), and the allowed minimum and maximum duration of each valued interval (by means of a function D associating to each value $v \in V$ a pair of time values). Moreover, the values of different state variables may be linked by (so-called) *synchronization rules*, requiring that, for every time interval where a given state variable x has the value v, there exist other time intervals where either the same or other state variables assume some given values, which are related by some temporal relation. All these constraints are specified in the *domain specification*. The planning process aims at synthesizing temporal flexible plans that satisfy the domain constraints and fulfill some given goals.

The evolution of a single temporal feature over a temporal horizon is called the timeline of that feature. In general, plans synthesized by temporal P&S systems may be temporally flexible. They are made up of flexible timelines, describing transition events that are associated with temporal intervals (with given lower and upper bounds), instead of exact temporal occurrences. In other words, a flexible plan describes an envelope of

possible solutions aimed at facing uncertainty during actual execution. In this regard, they can be exploited by an executive system for robust execution. Some formalizations have been recently proposed to describe timelines and timeline-based plans [13,14].

Broadly speaking a timeline consists of a sequence of values that are temporally qualified by means of *tokens*. A token specifies the temporal interval (start and end times) assigned to a specific value over a timeline. *Flexible tokens* specify a flexible interval for values over timelines (i.e. values have a start interval and end interval, instead of time points) and, as proposed in [13], they can be defined as follows (where $\mathbb{T}$ is the set of "time points" and $\mathbb{T}^\infty$ denotes $\mathbb{T} \cup \{\infty\}$):

Definition 1 *Let $x = (V, T, D)$ be a state variable describing the set of allowed values V, the value transition function T and the value duration function D for a domain feature. A flexible token for the state variable x is a tuple of the form*

$$(x^i, v, (b, b'), (e, e'), (d, d'))$$

where $i \in \mathbb{N}, b, b', e, e', d \in \mathbb{T}, d' \in \mathbb{T}^\infty, v \in V, b < e'$ and $d_{min} \leq d < d' \leq d_{max}$, where $(d_{min}, d_{max}) = D(v)$. The element x^j is the token identifier.

Definition 2 *A flexible timeline for x is a finite sequence of flexible tokens for x, whose identifiers are $x^0, x^1, ..., x^k$:*

$$\begin{aligned} FTL_x = (x^0, v_0, (b_0, b'_0), (e_0, e'_0), (d_0, d'_0)) \\ ...(x^k, v_k, (b_k, b'_k), (e_k, e'_k), (d_k, d'_k)) \end{aligned}$$

where $b_0 = b'_0 = 0$, $e_k = e'_k = H$ is the temporal horizon of the timeline and for all $i = 0, ..., k-1$, $e_i = b_{i+1}, e'_i = b'_{i+1}$ and $v_{i+1} \in T(v_i)$.

A flexible token represents the set of its instances, i.e. the set of all non-flexible tokens that satisfy the value duration constraints. Similarly a *flexible timeline* FTL_x represents the set of its instances. Namely, an instance of a *flexible timeline* FTL_x is made up of a sequence of instances of the tokens of FTL_x and an instance of a set **FTL** of timelines is a set of instances of the timelines in **FTL**.

However the representation of *flexible plans* must include also information about the relations that have to hold between tokens in order to satisfy the synchronization rules of the planning domain. Thus, the representation of flexible plans must include also a set of temporal relations on tokens, guaranteeing that such rules are satisfied.

Let us consider two flexible tokens, with token identifiers x^i and y^k, belonging respectively to the state variables x and y. A synchronization rule of the domain may require that the flexible token x^i precedes the flexible token y^k. In such a case the set of temporal relations of included in the flexible plan must contain the relations $x^i \text{ before } y^k$ (i.e. $e^{'}_x < b_y$) in order to satisfy the synchronization rule of the domain.

Like in [15], a small set of primitive temporal relations can be considered, in terms of which all the usual quantitative constraints can be defined, such as, for instance, the relations $x^i \text{ contains}_{[lb_1,ub_1][lb_2,ub_2]} y^k$ and $x^i \text{ overlaps}_{[lb_1,ub_1][lb_2,ub_2]} y^k$ (where $lb_j \in \mathbb{T}$ and $ub_j \in \mathbb{T}^\infty$, for $j = 1, 2$) .

Definition 3 *A flexible plan Π over the horizon H is a pair (**FTL**, R), where **FTL** is a set of flexible timelines over the same horizon H and R is a set of relations on tokens, involving token identifiers in some timelines in **FTL**.*

A flexible plan represents the set of its instances. An instance of a plan Π=(**FTL**, R) is an instance of **FTL** that respects the relations in R. When there are different ways how a synchronization rule can be satisfied by the same set of flexible timelines **FTL**, each flexible plan represents a choice among them, and different plans with the same set **FLT** and different sets of relations R represent different ways to satisfy synchronization rules.

A planner like those described in the following can leave timelines *open* on the right, i.e. they can leave an *undefined* temporal interval at the end of a timeline. This means that the planner does not *decide* which is the temporal evolution of the timeline after its last meaningful token.

3 Characterizing Robustness for Timeline-based Plans

Given a *flexible timeline-based planner* it is important to define some metrics that allow to characterize the capacity of the generated plans to absorb temporal deviations, i.e. their *robustness*. In this section, some quality metrics concerning temporal features of plans are introduced.

There are several works in the literature that define quality metrics for evaluating plans and planning algorithms [8,16,17,18]. In this regards we consider temporal metrics such as *fluidity* and *disruptibility* (introduced by [9] to characterize the *robustness* of schedules on temporal networks), in order to check temporal flexibility and provide an assessment of the robustness of timeline-based plans. In particular we adapt fluidity and disruptibility metrics to timelines and define the notion of the *makespan* of a timeline to indicate the "useful" portion of a timeline. Note that the term "makespan" is typically used in scheduling problems to express the maximum duration of a schedule. Here, we are using the same term with a slightly different meaning.

The *timeline fluidity* metric is an estimate of the capacity of the timeline to absorb temporal deviations w.r.t. other timelines. It is defined as follows:

Definition 4 *If FTL_x is a flexible timeline for the state variable x, the fluidity of the timeline w.r.t. the other timelines FTL_y of the plan is*

$$\xi(FTL_x) = \sum_{x^i \in FTL_x, y^j \in FTL_y} \frac{\rho(x^i, y^j)}{H \times n \times (n-1)} \times 100$$

where x^i is a token in the timeline FTL_x, y^j is a token in a timeline $FTL_y \neq FTL_x$, H is the temporal horizon of the plan, and n is the number of tokens involved in the computation.

Fluidity is computed by taking into account the temporal slack between tokens, that is a measure of the temporal flexibility between the end time of a token and the start time of another one:

$$\rho(x^i, y^j) = |d_{max}(x^i_{end}, y^j_{start}) - d_{min}(x^i_{end}, y^j_{start})|$$

where d_{max} and d_{min} are respectively the maximum and minimum allowed temporal distances between the end time of x^i and the start time of y^j.

The higher is the value of the *fluidity* of a timeline FTL_x, the higher is the capacity of other timelines of the plan to absorb its temporal deviations. Namely the higher is the value, the lower is the risk of cascading changes on other timelines of the plan. This metric provides a measure of temporal dependencies among the timelines of the plan.

The *timeline disruptibility* metric measures the amount of changes in a plan caused by the introduction of a delay in a timeline and is defined as follows:

Definition 5 *If FTL_x is a flexible timeline for the state variable x, the disruptibility of the timeline w.r.t. other timelines FTL_y of the plan is*

$$\psi(FTL_x) = \frac{1}{n} \sum_{x^i \in FTL_x, y^j \in FTL_y} \frac{\rho(0,x^i)}{|\{y^j : y^j \in \Delta(x^i,\delta)\}|}$$

where x^i and y^j are token of the timelines FTL_x and FTL_y, respectively. $\Delta(x^i, \delta)$ is the set of tokens in the plan that change after the introduction of a delay δ on the token $x^i \in FTL_x$.

Disruptibility is computed by taking into account the slack $\rho(0, x^i)$ of tokens, which is a measure of the temporal flexibility between the temporal origin of the timeline and the start time of the token.

$$\rho(0, x^i) = |d_{max}(0, x^i_{start}) - d_{min}(0, x^i_{start})|$$

It is a measure of the flexible temporal allocation of the token over the timeline.

The disruptibility metric $\psi(FTL_x)$ counts the number of changes (temporal deviations) in the plan due to a temporal delay on tokens x^i of the timeline FTL_x.

Finally the *makespan* metric of a timeline is a measure of the temporal flexibility between the last (meaningful) token of a timeline and the temporal horizon, when the timeline leaves an undefined temporal interval at its end.

Definition 6 *Given a flexible timeline FTL_x for the state variable x with k tokens, the makespan of the timeline is*

$$\mu(x) = \frac{\rho(x^k, H)}{H} \times 100$$

where $\rho(x^k, H)$ is the slack between the last token of the timeline $x^k \in FTL_x$ and the horizon H

$$\rho(x^i, H) = |d_{max}(x^i_{end}, H) - d_{min}(x^i_{end}, H)|$$

It is a measure of the portion of timeline left to be used.

The higher is the value of $\mu(FTL_x)$, the larger is the width of the flexible distance between the last token of FTL_x and the horizon.

In the next section we consider a case study in order to make an experimental evaluations of the different planners we have defined by means of our timeline-based planning

framework EPSL. In particular we aim at characterizing qualities of plans generated using different planner configurations. It is also interesting to evaluate relations among the defined metrics as, in some cases, metrics may be in contrast. This means that it is not possible to obtain a plan with the maximum level of all desired qualities but that the planner must be carefully configured in order to obtain the desired balance among all desired qualities

4 Extensible Planning and Scheduling Library

EPSL [12] is a layered framework built on top of APSI-TRF[1] [7]. It aims at defining a flexible software environment for supporting the design and development of timeline-based applications. The key point of EPSL flexibility is its interpretation of a planner as a "modular" solver which combines together several elements to carry out its solving process.

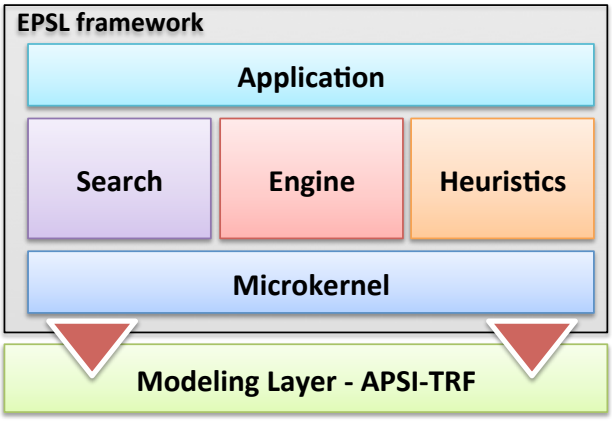


Fig. 1. EPSL Architectural Overview

Figure 1 describes the main architectural elements of the architecture of EPSL. The *Modeling layer* provides EPSL with timeline-based representation capabilities to model a planning domain in terms of *timeline*s, *state variable*s, *synchronization*s and manage *flexible plan*s. The *Microkernel layer* is the key element which provides the framework with the needed flexibility to "dynamically" integrate new elements into the framework. It is responsible to manage the lifecycle of the solving process and the elements composing the application instances (i.e. the planners). The *Search layer* and the *Heuristics layer* are the elements responsible for managing strategies and heuristics that can be used during the solving process.

[1] APSI-TRF is a software framework developed for the European Space Agency by the Planning and Scheduling Technology Laboratory at CNR (in Rome, Italy).

The *Engine layer* is the element responsible for managing the portfolio of algorithms, called *resolver*s, available to EPSL-based planners. Resolver encapsulate the logic for refining timeline-based plans. Each resolver, solves some specific conditions (called *flaws*) that threat the completeness or correctness of a plan. The set of available resolvers characterizes the expressiveness of the framework. Namely they determine what EPSL-based planners can actually do to solve problems.

Finally the *Application layer* is the top-most element which carries out the solving process and finds a solution if any. EPSL architecture allows to define modular planner instances which can be configured in different ways according to the particular feature of the problem to address. An EPSL-user configure planners by selecting the elements (e.g. heuristics, strategies and resolvers) that compose the application instance. Similarly the user can also extend EPSL capabilities by integrating domain-specific elements.

EPSL solving approach is a standard plan refinement search procedure which can be adapted to the particular problem to solve by changing the the planner configuration. As a matter of fact the particular strategy or heuristic applied can strongly affect planner behaviour and performances. In particular, the planner has two important choice points during a the search: (i) *node selection* choice concerning the selection of the search space node to expand next and (ii) *flaw selection* choice concerning the selection of the flaw to solve next in order to refine the current plan (i.e. node expansion).

We focused our attention on the *flaw selection* choice which is not a backtracking point of the search but, in our experience, a crucial aspect to enhance planner performances. Indeed, a careful selection of the next flaw to solve can *prune* the search space by cutting off branches that would not lead to solutions. Flaws can have dependencies, indeed, and the resolution of a flaw can simplify the resolution of other flaws in the plan. In this regard, EPSL allows to define *heuristic*s that support the search in selecting the most promising flaws to solve. Typically these *heuristic*s encapsulate some evaluation criteria that allow to rate plan flaws by taking into account one or more feature. In [11], we developed a domain independent heuristic, called *Hierarchical Flaw Selection heuristic* (HFS), which rate flaws by analyzing the *hierarchical* structure of a timeline-based domain.

4.1 Hierarchical Flaw Selection heuristic

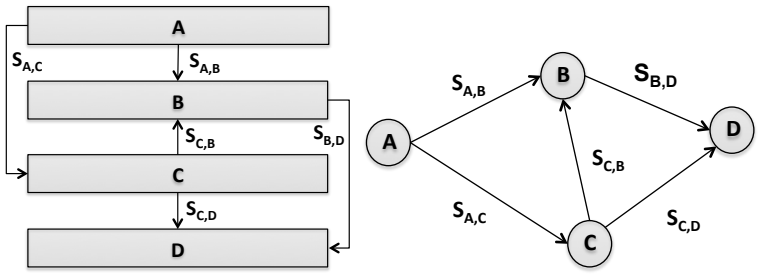
A timeline-based domain specifies relations between different timelines by means of *synchronization rule*s. Thus, given a timeline A and a timeline B, a synchronization rule $S_{A,B}$ from a token $x \in A$ to a token $y \in B$ implies a *dependency* between these timelines. Namely, tokens on timeline B are subject to tokens on timeline A.

Therefore, it is possible to build a *Dependency Graph* (DG) among timelines by looking at synchronization rules. Figure 2(a) shows a set of timelines with *synchronization rules* and Figure 2(b) shows the resulting dependency graph (note that the dependency graph defined here is different from the graph used in EUROPA2 [19]).

The DG encodes dependencies among timelines, and a hierarchy can be extracted by analysing the graph. An edge from a node A to a node B in the DG represents a dependency between timeline A and timeline B (i.e. tokens of timeline B depend on tokens of timeline A). Consequently the hierarchy level of timeline A is not lower than

the hierarchy level of B. If no path in the DG connects B to A, then A is at a higher level in the hierarchy than B (i.e. timeline A is more independent than timeline B). Conversely if A is connected to B and vice-versa in the DG (i.e. a cycle is detected) then timelines A and B have the same hierarchical level, and they are said to be hierarchically equivalent. For instance the hierarchy extracted from the DG in Figure 2 is $A \prec C \prec B \prec D$.

Usually planning domain specifications that follow a hierarchical modeling approach (like the approach described in [11]), generate a non-flat hierarchy of timelines (and sometimes even an acyclic DG).



(a) (b)

Fig. 2. From Synchronization rules to Dependecy Graph: (a) domain timelines and synchronization rules; (b) the dependency graph resulting from synchronization rules between timelines

The HFS exploits this hierarchy to define a *flaw hierarchy feature* and characterize the *independency* degree of plan flaws. The idea is to solve first "independent" flaws, i.e. flaws belonging to the top most timeline in the hierarchy (e.g. flaws on timeline A w.r.t. Figure 2), in order to *simplify* the resolution of "dependent" flaws. In addition to the hierarchy feature, HFS uses a *flaw type feature* to define a structure for the solving process and the *flaw degree feature* to characterize the *criticality* of a flaw (similarly to the *fail first principle* in constraint satisfaction problems).

The HFS selects the best flaw to solve next by combining together all the features described above as a *pipeline* of *filter*s:

$$\Phi^0(\pi) \xrightarrow{f_h} \Phi^1(\pi) \xrightarrow{f_t} \Phi^2(\pi) \xrightarrow{f_d} \Phi^3(\pi) \rightarrow \phi^* \in \Phi^3(\pi)$$

where f_h filters plan flaws according to the *flaw hierarchy* feature (i.e. it returns only the subset of flaws belonging to the most independent timeline of the hierarchy), f_t filters flaws according to the *flaw type* feature and f_d filters flaws according to the *flaw degree* feature. Then, given a set of flaws of a plan $\Phi^0(\pi)$ every filter extracts the subset of the relevant flaws according to the related feature. The pipeline resulting set $\Phi^3(\pi) \subseteq \Phi^0(\pi)$ is composed by flaws representing *equivalent choices* from the heuristic point of view, so HFS randomly select the "best" one to solve next $\phi^* \in \Phi^3(\pi)$.

5 Experimental Evaluation

The objective of the experimental analysis is to evaluate flexible timeline-based plans generated by EPSL-based planners with different configurations. In particular we evaluate plans by considering the *fluidity* and *makespan* metrics[3] as well as the planners time performances. In this regards, we use two configurations of EPSL-based planners by selecting two different *heuristic* functions: (i) the *HFS* planner, using the HFS heuristic; (ii) the *TFS* planner (*Type Flaw Selection* planner), which uses a heuristic function based only on the *flaw type* feature to select flaws (configuration used before the introduction of the HFS heuristc).

5.1 A pilot plant

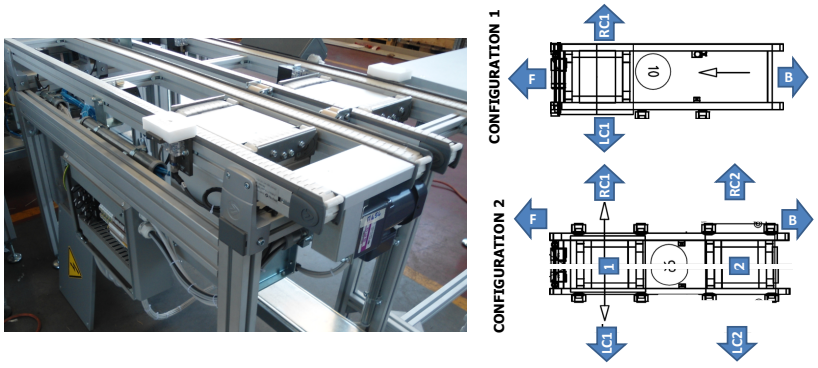
Here, we consider a pilot plant involved in an on-going research project called *Generic Evolutionary Control Knowledge-based mOdule* (GECKO): a manufacturing system for Printed Circuit Boards (PCB) recycling [20]. The objective of the system is to analyze defective PCBs, automatically diagnose their faults and, depending on the gravity of the malfunctions, attempt an automatic repair of the PCBs or send them directly to shredding.

The pilot plant contains 6 working machines (M1,..., M6) that are connected by means of a Reconfigurable Transportation System (RTS), composed of mechatronic components, i.e., transport modules. Figure 3(a) provides a picture of a transport module. Each module combines three transportation units. The units might be either unidirectional or bidirectional; specifically the bidirectional units enable the lateral movement (i.e., cross-transfers) between two transportation modules. Thus, each transport module can support two main (straight) transfer services and one to many cross-transfer services. Figure 3(b) depicts two possible configurations.

Configuration 1 supports the forward (F) and backward (B) transfer capabilities as well as the left (LC1) and right (RC1) cross transfer capabilities. Configuration 2 extends Configuration 1 by integrating a further bidirectional transportation unit with cross transfer capabilities LC2 and RC2. The maximum number of bidirectional units within a module is limited just by its straight length (three, in this particular case). The transport modules can be connected back to back to form a set of different conveyor layouts. The manufacturing process requires PCBs to be loaded on a fixturing system (pallet) in order to be transported and processed by the machines. The transportation system is to move one or more pallets and each pallet can be either empty or loaded with a PCB to be processed.

Such an RTS generally allows for a number of possible routing solutions for each single pallet which is associated to a given destination. Transport modules control systems have to cooperate in order to define the paths the pallets have to follow to reach their destinations. These paths are to be computed at runtime, according to the actual status and the overall conditions of the shop floor (i.e., no static routes are used to move pallets).

[3] In this preliminary experimental analysis, *disruptibility* metric does not provide relevant data



(a) (b)

Fig. 3. (a) A transport module; (b) Their transfer services

The description of the distributed architecture and some experimental results regarding the feasibility of the distributed approach w.r.t. the part routing problem can be found in [21]. *Transportation Module*s (TMs) rely on P&S technology to synthesize activities for supporting the work flow within the shop floor. As a matter of fact TM agents are endowed with Timeline-based planners (build on top of EPSL framework) that assess modules' internal capabilities during the part routing computation process and build modules' plans for coordination and transportation task.

Figure 4 shows the timeline-based model of a generic *transportation module* (TM) of the GECKO case study extended with energy consumption resource to estimate energy consumption profile of transportation tasks. The *Channel* state variable models the high-level transporting tasks the TM is able to perform. Each value of the *Channel* state variable models a particular transportation task indicating the ports of the module involved in the execution of the task. For instance *Channel_F_B* models the task of transporting a pallet form port F to port B w.r.t. Figure 3(b).

The *Change-Over* state variable models the set of internal configuration the transportation module can assume in order to actually exchange pallets with other modules. Namely configurations identify the internal paths a pallet can follows to traverse the module. For instance, *CO_F_B* in Figure 4 represents the configuration needed to transport a pallet from port F to port B.

The *Energy-Consumption* resource models the *energy consumption* policy o the TM. It estimates the energy consumption of transportation tasks of the module, i.e. the energy requirements for channel activities of the module. Activities such as moving conveyors or cross transfers require energy to be performed and we model a system requirement which entails that the instant energy consumption of TMs cannot exceed a predefined limit for safety and optimization reasons of the physical device. In this way, the planner must organize activities in order to not violate the energy consumption constraint of the module.

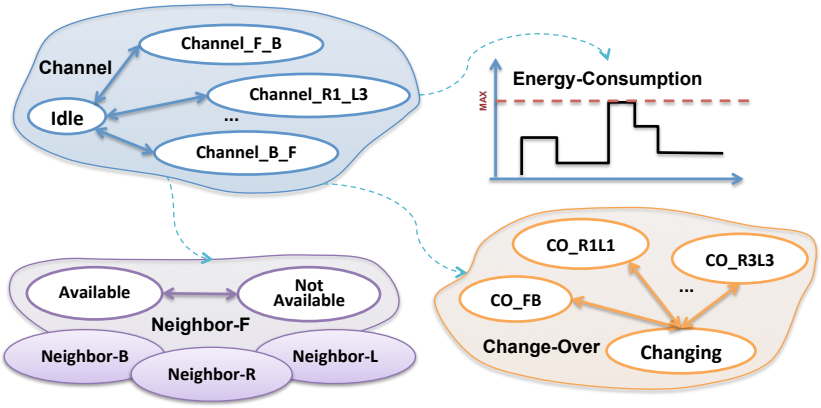


Fig. 4. Timeline-based model for a full instantiated TM

A set of *external state variable*s complete the domain by modeling possible states of TM's neighbor modules. Neighbors are modeled by means of external state variables because they are not under the control of the module. Namely, the TM cannot decide the state of its neighbors. However it is important to *monitor* their status because a TM must cooperate with them in order to successfully carry out its tasks. For instance the TM must cooperate with *Neighbor_F* and *Neighbor_B* to successfully perform a *Channel_F_B* task. Therefore *Neighbor_F* and *Neighbor_B* must be *Available* during task "execution".

Finally a set of *synchronization rule*s specify how a TM *implement*s its *channel* tasks. Figure 4 groups these rules in the dotted arrows between state variables for readability reasons. These rules specify *operative* constraints (causal and temporal constraints) describing the sequence of internal configurations and "external" conditions needed to safely perform *Channel* tasks. For instance, a synchronization rule for the *Channel_F_B* task require that the module must be set in configuration *CO_F_B* and that neighbor F and neighbor B must be *Available* during the "execution" of the task.

5.2 Evaluating Plan Robustness

In the GECKO case study we have defined four planning domain variants by considering different physical configurations of a TM of the manufacturing plant, i.e., by varying the number of cross-transfer units composing the module and under the assumption that module's neighbors are always available for cooperation. Namely, the tests were run on the following planning domains: (i) *simple*, the configuration with no cross transfer; (ii) *single*, the configuration with only one cross transfer unit; (iii) *double*, the configuration with two cross transfer units; (iv) *full*, the configuration with three cross transfer units (the maximum allowed in the case study we considered). The higher is the number of available cross transfers, the higher is the number of elements and constraints the planner has to deal with at solving time.

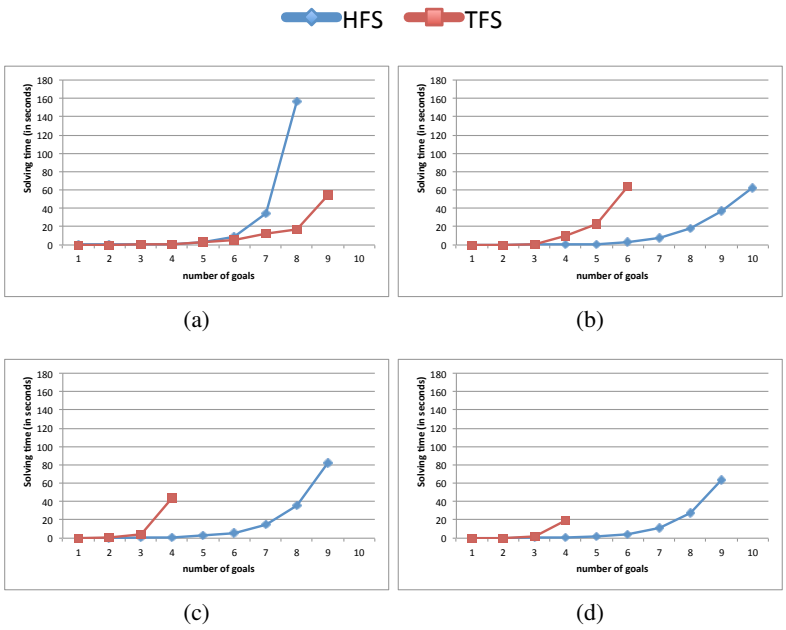


Fig. 5. *HFS* and *TFS* planners performances on: (a) *simple* configuration; (b) *single* configuration; (c) *double* configuration; (d) *full* configuration

The charts in Figure 5 show the solving time trends of the EPSL-based planners (within a timeout of 180 seconds) w.r.t. the growing dimension of the planning problem (i.e. a growing number of goals) and the growing complexity of the module to control (i.e. the number of available cross transfers). The results show that the *HFS* planner dominates *TFS* planner on the considered planning domains. The introduction of the HFS heuristic in the solving process entails a general improvement of the performances in terms of both solving time and scalability of framework.

Let us consider the subset of problems solved by both *HFS* and *TFS* planners in Figure 5 and make a comparison of generated plans. Figure 6 compares the generated plans by considering the *fluidity* metric previously introduced.

The chart in Figure 6(a) shows the trend of the fluidity of the plans w.r.t. the growing complexity of the addressed problems in terms of number of goals and constraints to manage. Results show that the higher is the complexity of the problems the lower is the amount of fluidity of the generated plans. Thus, as expected, the higher is the number of tokens on timelines the higher is the probability that a temporal deviation on a token causes temporal deviations on tokens of other timelines.

The charts in Figure 6(b) and (c) compare respectively the total fluidity and the average makespan of the generated plans. Results show that *TFS* planner generates plans more robust than *HFS* planner w.r.t. fluidity metric. The total fluidity of the plans generated by *TFS* planner, indeed, is higher than the total fluidity of the plans generated by *HFS* planner. Thus, the introduction of HFS heuristic in the solving process seems to

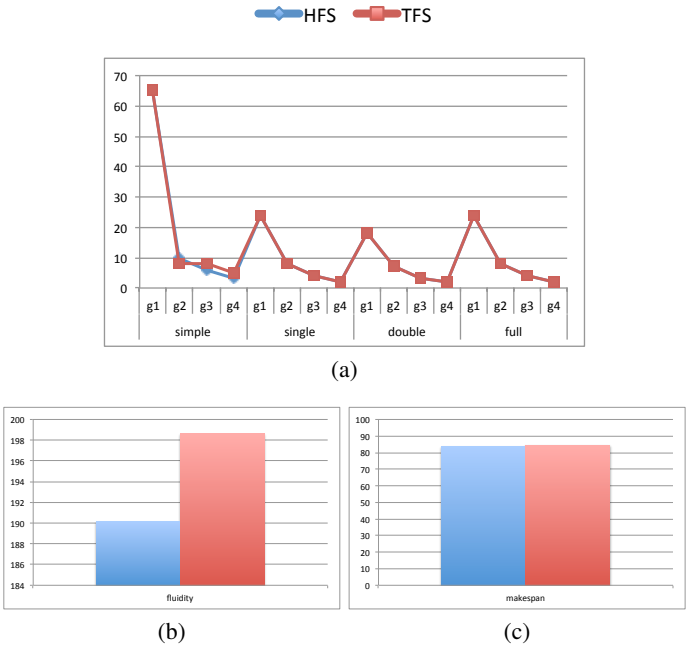
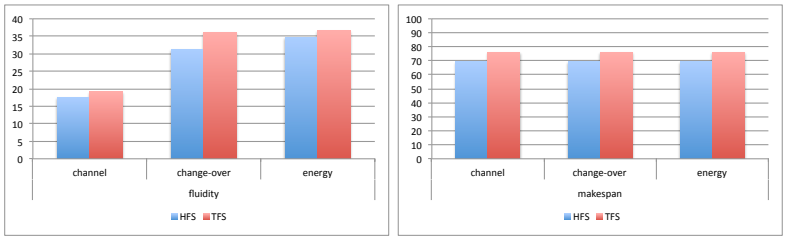


Fig. 6. Comparison of HFS and TFS planners on: (a) plan fluidity trend; (b) total plan fluidity; (c) average plan makespan

lead to a loss of the robustness of the generated plans despite a significant improvement of the solving capabilities as shown in Figure 5.

This can be explained by considering that the *TFS* planner maintains a "complete" view of the plan during the solving process. The planner reasons about the overall plan by taking into account all the timelines of the domain, so that it can organize the tokens on timelines as best as it can. Conversely, the *HFS* planner maintains a "local" view of the plan which is related to single timelines of the domain. Namely, the *HFS* planner builds one timeline at a time. Therefore, when the planner must manage "intermediate" timelines, its choices are partially constrained by the choices made before.

This behavior of HFS heuristic can lead also to a loss of performances in some specific cases such as the *simple* domain in Figure 5(a). The chart shows that, despite the general trend, the *TFS* planner performs better than *HFS* planner in the *simple* domain. Because of its "local" approach, the *HFS* planner is not able to effectively organize tokens on timelines as *TFS* planner does. In particular *TFS* planner is able to efficiently group tokens of *channel* timeline requiring the same configuration of the module, i.e. the same type of token on the *change-over* timeline. Conversely the *HFS* reasons on one timeline at a time, so it is not able to group tokens requiring the same configuration on *channel* timelines. As a consequence the planner must manage much more tokens on the *change-over* timeline increasing the complexity of the problem resolution.

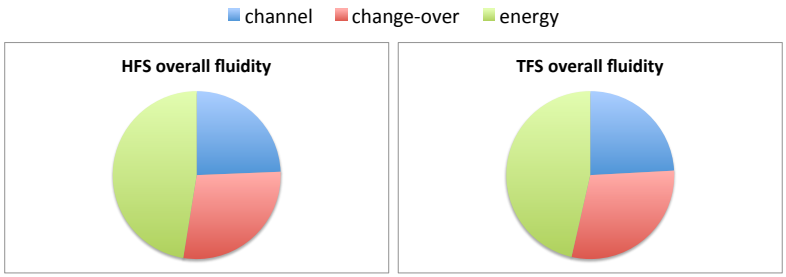


(a) (b)

Fig. 7. Comparison of plans generated for the *simple* domain on: (a) the total fluidity, (b) the average makespan

As a matter of fact the chart of Figure 7(a) shows that the fluidity of the *change-over* timeline in the *TFS* generated plans is quite higher than the fluidity of the same timeline in the *HFS* generated plans.

Finally the introduction and analysis of temporal metrics allow also to extract useful information about planning domains. Let us consider the contribution of single timelines to the total fluidity of the generated plans shown in Figure 8.



(a) (b)

Fig. 8. Contribution of the single timelines to the overall plan fluidity for (a) *HFS* generated plans and (b) *TFS* generated plans

It is possible to see that, regardless the heuristic applied during the solving process, the relationships among domain timelines remain constant. The charts in Figure 8, show that, in general, the *energy* timeline is the one with the highest value of fluidity while the *channel* timeline is the one with the lowest value of fluidity. Thus, the *channel* timeline is the most critical one w.r.t. the robustness of the plan. Namely a temporal deviation on a token of the *channel* timeline is more likely to produce side effects on other timelines of the domain than tokens on other timelines. This sort of information is very interesting since they can be used to classify planning domains and introduce some learning mechanism to adapt the solving process to the specific features of a planning domain.

6 Conclusions and Future works

In this paper we have presented our preliminary work about the introduction of quality metrics to evaluate the robustness of flexible timeline-based plans. In particular, we have adapted some temporal metrics defined in scheduling to timelines. Then we have introduced EPSL, a timeline-based planning framework we use to design and develop P&S applications. Finally we made an experimental evaluation of two EPSL-based planners on a manufacturing case study.

Experimental results have shown how temporal metrics allow to make an assessment of the heuristics applied during the solving process. In particular, temporal metrics let us able to evaluate EPSL-based planners both from the solving performance and the plan quality point of views. An important issue to be addressed in future works is the introduction of a "dynamic" measure of flexibility in order to make a deeper analysis and assessment of the robustness of a plan. In particular we believe that an analysis of this sort is especially needed to provide a valid estimate of the disruptibility metric. Another objective is to develop some parametrized search heuristics heuristcs allowing to exploit temporal features of (partial-)plans during the solving process.

Moreover the experimental results have shown that these metrics charaterize information about the structure and relationships among domain timelines. Thus, another interesting future goal is to introduce some learning mechanism for dynamically adapting heuristics to the specific features of the problem to address.

Acknowledgments. Andrea Orlandini is partially supported by the Italian Ministry for University and Research (MIUR) and CNR under the GECKO Project (Progetto Bandiera "La Fabbrica del Futuro").

References

1. Muscettola, N.: HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., ed.: Intelligent Scheduling. Morgan Kauffmann (1994)
2. Jonsson, A., Morris, P., Muscettola, N., Rajan, K., Smith, B.: Planning in Interplanetary Space: Theory and Practice. In: Proceedings of the 5th International Conference on AI Planning and Scheduling (AIPS). (2000)
3. Cesta, A., Cortellessa, G., Denis, M., Donati, A., Fratini, S., Oddi, A., Policella, N., Rabenau, E., Schulster, J.: MEXAR2: AI Solves Mission Planner Problems. IEEE Intelligent Systems **22**(4) (2007) 12–19
4. Ceballos, A., Bensalem, S., Cesta, A., de Silva, L., Fratini, S., Ingrand, F., Ocón, J., Orlandini, A., Py, F., Rajan, K., Rasconi, R., van Winnendael, M.: A Goal-Oriented Autonomous Controller for space exploration. In: Proceedings of the 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA). (2011)
5. Barreiro, J., Boyce, M., Do, M., Frank, J., Iatauro, M., Kichkaylo, T., Morris, P., Ong, J., Remolina, E., Smith, T., Smith, D.: EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In: ICKEPS 2012: the 4th Int. Competition on Knowledge Engineering for Planning and Scheduling. (2012)
6. Ghallab, M., Laruelle, H.: Representation and control in ixtet, a temporal planner. In: AIPS. Volume 1994. (1994) 61–67

7. Cesta, A., Fratini, S.: The Timeline Representation Framework as a Planning and Scheduling Software Development Environment. In: PlanSIG-08. Proc. of the 27^{th} Workshop of the UK Planning and Scheduling Special Interest Group, Edinburgh, UK, December 11-12. (2008)
8. Roberts, M., Howe, A., Ray, I.: Evaluating diversity in classical planning. In: Proceedings of the 24th International Conference on Planning and Scheduling (ICAPS). (2014)
9. Policella, N., Smith, S.F., Cesta, A., Oddi, A.: Generating robust schedules through temporal flexibility. In: ICAPS. Volume 4. (2004) 209–218
10. Cesta, A., Oddi, A., Smith, S.F.: Profile-based algorithms to solve multiple capacitated metric scheduling problems. In: AIPS. (1998) 214–223
11. Umbrico, A., Orlandini, A., Cialdea Mayer, M.: Enriching a timeline-based planner with resources and hierarchical reasoning. In: Proceedings of the 14th Conference of the Italian Association for Artificial Intelligence (AIxIA). (2015) To appear.
12. Cesta, A., Orlandini, A., Umbrico, A.: Toward a general purpose software environment for timeline-based planning. In: 20th RCRA International Workshop on" Experimental Evaluation of Algorithms for solving problems with combinatorial explosion. (2013)
13. Cialdea Mayer, M., Orlandini, A., Umbrico, A.: A formal account of planning with flexible timelines. In: The 21st International Symposium on Temporal Representation and Reasoning (TIME), IEEE (2014) 37–46
14. Cimatti, A., Micheli, A., Roveri, M.: Timelines with temporal uncertainty. In: AAAI. (2013)
15. Cialdea Mayer, M., Orlandini, A.: An executable semantics of flexible plans in terms of timed game automata. In: The 22st International Symposium on Temporal Representation and Reasoning (TIME). (2015) To appear.
16. Benton, J., Coles, A.J., Coles, A.: Temporal planning with preferences and time-dependent continuous costs. In: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS). Volume 77. (2012) 78
17. Smith, D.E.: Choosing objectives in over-subscription planning. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS). Volume 4. (2004) 393
18. Nau, D., Ghallab, M.: Measuring the performance of automated planning systems. Technical report, DTIC Document (2004)
19. Bernardini, S., Smith, D.E.: Towards search control via dependency graphs in europa2 (2010)
20. Borgo, S., Cesta, A., Orlandini, A., Rasconi, R., Suriano, M., Umbrico, A.: Towards a cooperative -based control architecture for a reconfigurable manufacturing plant. In: 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2014), IEEE (2014)
21. Carpanzano, E., Cesta, A., Orlandini, A., Rasconi, R., Valente, A.: Intelligent dynamic part routing policies in plug&produce reconfigurable transportation systems. CIRP Annals - Manufacturing Technology **63**(1) (2014) 425 – 428