Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1999), *13*, 91–104. Printed in the USA.

Temporal reasoning in process planning*

MIRA BALABAN AND DAN BRAHA

Department of Industrial Engineering and Management, Ben-Gurion University, P.O.B. 653, Beer-Sheva 84105, Israel

(RECEIVED October 6, 1998; REVISED October 23, 1998; ACCEPTED November 4, 1998)

Abstract

Computer-aided process planning has been recognized as an important tool for coordinating the different operations involved in making the product. While temporal knowledge is central to the design of efficient and reliable process plans, little attention is given to the integration of process planning and temporal processing and reasoning. To fill the void, we propose in this paper a practical approach, which is inspired by the framework of Temporal Constraint Satisfaction Problem (TCSP), to integrate process planning and temporal reasoning. We show that a TCSP formulation is a subset of a formulation using a reified temporal logic, and discuss the advantages of using such a restricted model. To reflect more realistic process planning encountered in real manufacturing environments, we present a model, called *n*-TCSP, which is a generalization of the TCSP framework. We envision the proposed temporal reasoning framework as one of the modules in the evolving new intelligent computer-aided process planning.

Keywords: Temporal Reasoning; Temporal Constraint Satisfaction; Process Planning; Computer-Aided Process Planning

1. INTRODUCTION

Process planning is an activity within a manufacturing system that selects the sequence of operations and production processes needed to convert a part from one form to another. The initial specification is usually given in the form of a predefined engineering diagram. The processes and operations usually include tooling, fixtures and machinery, and the final form usually include dimensions, shape, tolerances, and surface finish desired (Chang & Wysk, 1985; Nevins & Whitney, 1989). Because this task is highly labor intensive and time consuming, Computer-aided Process Planning (CAPP) tools were developed as an important link between computer-aided design (CAD) and computer-aided manufacturing (CAM).

Much work is devoted to the generation of production and assembly sequences from a product design (DeFazio & Whitney, 1987; Lin & Chang, 1991; Woo & Dutta, 1991). One approach to the problem creates a precedence diagram (in terms of operations) from a basic design. The process planner then selects production processes and resources (e.g., material-handling equipment and machines), and assigns operations to the selected production processes. Nevins and Whitney (1989) suggest that, as part of the concurrent engineering process, all possible production and assembly sequences should be considered during the design phase.

The dominating approach in CAPP is to formulate the problem of selecting an assembly sequence as a cost minimization problem (e.g., Schmidt & Jackman, 1995). For that purpose, the problem is simplified by assuming that process duration times are known with certainty, and each production process can be associated with a single time factor standing for its duration. However, in practice, only lower and upper bounds on process duration are usually known. A more realistic approach is to capture such uncertainty by associating processes with *time intervals* that include all of their possible duration times. That is, for each process, its beginning and ending time points can be modeled explicitly, and the possible duration times can be captured by the temporal distances that are allowed between these two points. The following examples demonstrate this idea:

1. The temporal information "the computer numerically *controlled* (CNC) lathe cut off the part within 1 to 2 min" means that the time interval between the ending and the beginning time points of this process is $[1, 2]$.
2. The temporal information "the operation 'turn square groove' can be performed either by a lathe machine

*This work was supported in part by the Paul Ivanir Center for Robotics and Production Management at Ben-Gurion University of the Negev.

Reprint requests to: Mira Balaban, Department of Industrial Engineering and Management, Ben-Gurion University, P.O.B. 635, Beer-Sheva 84105, Israel.

within 1 to 2 min *or* by a boring mill machine within 3 to 4 min," states that the permitted duration times lie either within the interval $[1,2]$ or within the intervals $[3,4]$. This temporal constraint can be represented by associating the operation 'turn square groove' with the set of time intervals $\{[1,2], [3,4]\}$. That is, the alternative possibilities are captured by a *disjunctive constraint*.

3. The temporal constraint "the horizontal mill starts to turn the right chamfer at least 2 min after the lathe finishes cutting off the part"; states that the time interval between the beginning time point of the 'turn the right chamfer' process and the ending time point of the 'cutting off the part' process is $[2,\infty]$.
4. The temporal information "the surface mount assembly of the resistors starts after the paste is applied, but has to be completed before the paste becomes dry—otherwise, the (printed circuit board) PCB is scraped." means that if it is known that the paste becomes dry within t min, then the time interval between the ending time points of the 'surface mount assembly of the resistors' process and the 'apply paste' process is $[0,t]$.
5. The temporal statement "if the CNC lathe #1 finished its operation ('turn square groove') on Part #2 while Part #1 awaits for its second operation ('turn left chamfer'), then the CNC lathe #1 can be used to turn the left chamfer. Otherwise, use the CNC boring mill #1 to turn the left chamfer." presents a temporal constraint involving more than two time points. The constrained time points are t_{1e}, the ending time point of the first operation on Part #1, t_{2e}, the ending time point of the 'turn square groove' operation of the CNC lathe #1 on Part #2, t_b, the beginning time point of the 'turn left chamfer' process, and t_{3e}, the ending time point of the most recent operation of the CNC boring mill #1. It states that either the time intervals from t_{2e} to t_{1e} and from t_{2e} to t_b are $[0,\infty]$, or the time intervals from t_{2e} to t_{1e} is $[-\infty,0]$ (i.e., t_{1e} occurs before t_{2e}) and the time interval from t_{3e} to t_b is $[0,\infty]$. This constraint can be written, in short, *as*$\{([0,\infty]_{t_{2e},t_{1e}}$ and $[0,\infty]_{t_{2e},t_b})$, $([-\infty,0]_{t_{2e},t_{1e}}$ and $[0,\infty]_{t_{3e},t_b})\}$.

Given such temporal information, we aim at deriving answers to queries and tasks such as: (1) "Is the given temporal knowledge consistent?"; (2) "Is it possible that the assembly sequence ends at time t?"; (3) "What are the possible times at which the surface mount assembly can be performed?" In addition, it is desirable to generate one or more assembly sequence scenarios (that is, selecting production processes and resources for different assembly operations) consistent with the temporal information provided, finding all feasible times that a given assembly operation can be performed, and finding all possible relationships (in terms of temporal constraints) between two given assembly operations. Ultimately, an effective procedure to the problem of selecting an optimal assembly sequence (e.g., in terms of cost) can find the global optimal among all feasible assembly scenarios generated by the temporal reasoning system.

The above discussion illuminates the relevance and importance of temporal processing for process planning. Nevertheless, in current CAPP little attention is given to this subject. In this paper we embark on the task of integrating temporal reasoning into this paradigm. As a first step, we consider a deliberately limited and computationally decidable temporal model, that can account for disjunctive temporal knowledge of the kind demonstrated above. The model, termed *Temporal Constraint Satisfaction Problems* (*TCSP)*, was first introduced in (Dechter et al., 1991). Its origins lie in the general school of constraint network formalisms, termed *Constraint Satisfaction Problems* (*CSP)*. [For more information, consult, for example, Montanari (1974), Mackworth (1977), Freuder (1978, 1982), Haralick and Elliott (1980), Nudel (1983), and Dechter and Pearl (1987)]. The following example illustrates how the TCSP model can be used to reason about temporal information in process planning.

EXAMPLE 1. Consider the material flow in an assembling environment, which has a single product type and four workstations. The product has an upper and a lower part that has to be assembled. The assembly is performed through three stages of operations. In the first stage, a solder paste is applied to the upper and lower parts within 8 to 10 s. In the second stage, screws are fastened to the parts either with the Pneumatic screw driver within 10 to 17 s or with the Vertical Mill within 7 to 9 s. In the third stage, the ERV7 Robot picks the sealing with the vacuum tool and carries the sealing to the part on the fixture within 12 to 17 s. The conveyor transports the parts from stage to stage within 13 s. The system requires an initial setup within 65 to 85 s. Because the applied paste (during stage 1) is usable for a limited amount of time, the second operation (assembling the upper and lower parts) and the third operation (sealing) have to be completed within 30 to 40 s and within 50 to 75 s after the paste is applied, respectively. If the second and third operations are not completed before the paste becomes dry, then the product is scraped. The total assembly time of the product is constrained to be within 120 to 130 s. ■

The process planner wishes to answer questions such as: (1) "Is the information provided by the process planner consistent? In particular, is it possible that the assembly process terminates as scheduled (within 120 to 130 s)?"; (2) "What are the possible times at which the upper and lower parts can be assembled?"; (3) "Is it possible that in a scenario, which is consistent with the information provided, the Pneumatic screw driver is used in the second stage?"

To answer the above queries, we single out the important operations in the assembling plan. The temporal information constrains their beginning and ending time points. The operation "setting up the assembly system" has a beginning and ending time points denoted by the variables X_{1s} and X_{1e}, respectively. The operation "applying a solder paste to the upper and lower parts" has beginning and ending variables X_{2s} and X_{2e}, respectively; the operation "fastening screws

to the upper and lower parts" has beginning and ending variables X_{3s} and X_{3e}, respectively; and the operation "sealing the product" has beginning and ending variables X_{4s} and X_{4e}, respectively. We also introduce a special time point, denoted X_0, to stand for the beginning of the assembly process.

The problem sets temporal constraints on these points. For example, the proposition "a solder paste is applied to the upper and lower parts within 8 to 10 s" implies that the temporal distance between X_{1s} and X_{1e} is constrained by $8 \leq X_{1e} - X_{1s} \leq 10$. The proposition "screws are fastened to the parts either with the Pneumatic screw driver within 10 to 17 s or with the Vertical Mill within 7 to 9 s" implies that $10 \leq X_{2e} - X_{2s} \leq 17$ or $7 \leq X_{2e} - X_{2s} \leq 9$. The proposition "the second operation has to be completed within 30 to 40 s after the paste is applied" implies that $30 \leq X_{2e} - X_{1e} \leq 40$. The fact that the conveyor transports the parts from the first stage to the second stage within 13 s, means that the temporal distance between X_{1e} and X_{2s} is constrained by $X_{2s} - X_{1e} = 13$. The constraint on the total assembly time of the product is: $120 \leq X_{3e} - X_0 \leq 130$.

Temporal constraint systems that use a set of unary and binary constraints, as in the above example, can be graphically described by representing time points by vertices and the propositions by labeled edges. Figure 1 presents the graphical description of the problem of Example 1.

Using algorithmic methods developed for TCSP networks, we can answer queries about the consistency of the temporal constraints, feasible temporal distances between time points, feasible time values for time points, and eventually, find solutions, that is, values for the time points. Indeed, for general disjunctive networks, the problem is NP-hard, and the algorithms use heuristics (Dechter et al., 1991), or approximate the solutions (Schwalb & Dechter, 1997), or solve special cases (Dechter et al., 1991; Balaban & Rosen, 1999).

Yet, the TCSP model is not a full-fledged logic and it does not support reasoning about change. For example, it cannot express a general inference statement such as "if the CNC lathe #1 breaks down while performing 'turn square groove' on Part #2, then Part #2 is also broken, but the rest of the process is unaffected. The problem can be fixed by inserting a new lathe machine to the process, and repeating the operation of the broken machine." General purpose reasoning about change lies in the core of artificial intelligence theories for representation and reasoning about common sense knowledge. Nevertheless, such theories are computationally undecidable; handling scenarios as the above one with a general inference theory is impractical. The TCSP model, in contrast, can account for a limited kind of temporal knowledge, and is meant to answer only particular queries. However, it is associated with much developed algorithms and heuristics for answering such queries. The next section includes a short account of temporal reasoning in AI. In particular, we describe recent efforts to incorporate temporal constraints into general-purpose reasoning without breaking the decidability bound. We also demonstrate how reified temporal logic can be used to capture the temporal content in a process-planning activity. We show that a TCSP network is a subset of such a formulation, for which known inference algorithms are available.

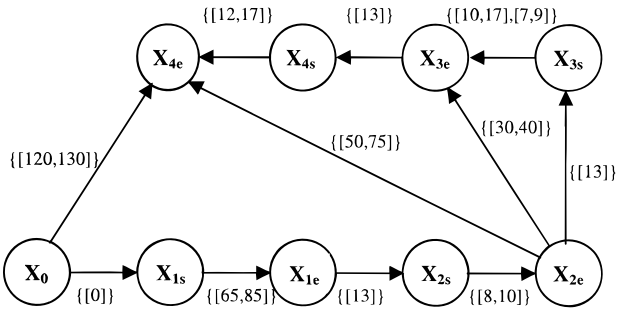

Fig. 1. A directed constraint graph of Example 1.

In this paper, we propose to use a TCSP network to integrate process planning with reasoning about temporal constraints. The TCSP network can be one of the modules in the evolving new intelligent CAPP systems. Moreover, we further generalize the standard binary TCSP with *n*-ary constraints. The generalization is motivated by the need to account for complex temporal information, such as when some resources (e.g., machines or robots) in the assembly system are flexible and capable of performing multiple operations; or when material handling tasks (such as placing a part on the conveyor, removing a part from a feeder, or inserting a part in a CNC machine) can be performed according to several scheduling procedures.

Section 2 includes a short account of temporal reasoning in AI, including a reformulation of Example 1, above, in a reified temporal logic. In Section 3, the relevant concepts and results for TCSPs are presented. In Section 4, we consider two examples of process planning. The examples suggest that the TCSP framework that deals with binary temporal constraints (between two time points) is too weak because in practical process planning the temporal constraints are set on several time points. To this end, we extend the TCSP model to include *n*-ary temporal constraints, and show that the binary TCSP model is a degenerate case of the proposed framework (called *n*-TCSP). Interestingly, the generalized TCSP model involves no additional computational cost. Finally, we use the *n*-TCSP model to formulate the temporal content of the previously introduced examples. Conclusions and future research directions are discussed in Section 5.

2. TEMPORAL REASONING IN ARTIFICIAL INTELLIGENCE

Temporal processing is central for common sense reasoning in real world scenarios pertinent to plan generation and recognition systems, applications that create explanations, and prediction systems. To reflect realistic common sense rea-

soning, temporal processing must represent and reason about changing situations, causal laws that determine such changes, occurrences of events and their effects, and constraints imposed on possible situations. In addition, temporal reasoning is used in deductive databases to express temporal dynamic constraints in the database subject domain, such as for periodical medical treatments (Chomicki, 1996). In software verification, temporal reasoning is used to guarantee desirable properties such as liveliness and fairness of concurrent processes (Manna & Pnueli, 1981). The dynamic logic of Pratt (1976) and Harel (1979), developed for the formal analysis of computer programs, also reasons about the changing situations involved in computer processes.

Common sense reasoning systems cope with the need to model changes and lack thereof over time. The main problem in common sense reasoning about time is the need to reason with incomplete information about the world and about the effects of actions. The *frame problem* (Pylyshyn, 1987) deals with the effects of actions, while the *qualification problem* (Shoham, 1988) deals with common knowledge that is usually not explicitly specified. In the artificial intelligence paradigm, the theories developed to deal with these issues usually build on extending first-order predicate logic with particular sorts to account for the dynamic nature of the problem. In the *situation calculus* (McCarthy & Hays, 1969; Baker, 1991), the time passage is modeled by introducing a *situation sort* where each situation stands for a point in time. Change over time is modeled by the sort of *fluents*, which are time varying terms (McDermott, 1982). Other approaches use explicit sorts for *time* (Shoham, 1988; Bacchus et al., 1991) or for *events* (Kowalski & Sergot, 1986; Vila & Reichgelt, 1996). Implementation issues for temporal knowledge bases that cope with common sense reasoning are discussed in (Dean & McDermott, 1987; Dean, 1989).

Most applications require, in addition to common sense reasoning capabilities, the ability to model the temporal ontology and its qualitative and quantitative properties. Allen (1983, 1984), in his seminal papers developed a qualitative temporal ontology based on *intervals*, rather than situations. The ontology is based on qualitative interval relationships such as being *before*, *after*, *meeting*, *during*, *contained*, *equal*, etc. Much research on solving and testing consistency of constraints in this ontology followed (Vilain & Kautz, 1986; Ladkin & Reinefeld, 1992; VanBeek, 1992; Golumbic & Shamir, 1993). Another temporal ontology that was independently investigated is that of metric constraints among time points, which denote the beginning or ending of events. This model, termed *Temporal Constraint Satisfaction Problems* (TCSP), was introduced in Dechter et al. (1991) and further investigated in Schwalb and Dechter (1997) and Balaban and Rosen (1999). The TCSP model generalizes the linear inequalities formalisms of Malik and Binford (1983) and of Valdes-Perez (1986). Combinations of the qualitative and the metric TCSP models are introduced in Meiri (1996) and Kautz and Ladkin (1991).

A large body of knowledge was accumulated for solving particular tasks in the qualitative and metric TCSP models. These tasks include (1) deciding consistency of a problem and finding a solution; (2) deciding entailment, that is, whether a given relation between intervals, between time points, or between a time point and an interval is entailed from the problem specification; and (3) enumerating all solutions and computing the minimal temporal constraints allowed by the problem specification.

Because the models are restricted to conjunctions and disjunctions of relations, all the above tasks are computationally decidable. Yet, the ontologies for solving temporal constraints have a limited expressive power. They cannot describe situations where the number of time points or intervals is not known in advance, such as periodic situations with unknown time period size. Another limitation is that they cannot express combinations of temporal and nontemporal knowledge. Recent efforts, discussed in Schwalb et al. (1997), propose a combination of *Datalog* (a restricted and decidable deductive databases language), with qualitative and metric temporal constraints. Schwalb et al. (1997) propose a decidable language, termed *Token-Datalog*, that can account for if–then rules, temporal constraints, and periodic constraints.

As mentioned above, we consider in this paper a deliberately limited and computationally decidable TCSP model as a first step to the integration of temporal reasoning and process planning. The TCSP model is appealing because (1) it can account for disjunctive temporal knowledge of the kind demonstrated above; (2) it supports and provides a framework for the representation and reasoning about quantitative and metric temporal information (i.e., sentences involving numerical time points and time differences between events), which is central to our process planning application; and (3) it is amendable to known algorithms developed by other network-based methods of constraint satisfaction (e.g., Montanari, 1974). Section 3 is devoted to the introduction of the temporal constraint satisfaction problem. In the following subsection (2.1) we formulate Example 1 in temporal logic. We use a reified temporal logic because this approach aims at reasoning about common sense temporal knowledge.

2.1. Using temporal logic for process planning—Example and discussion

Classical logic is appropriate for describing *static situations* because it is timeless. Assignment of truth values to propositions is independent from possible changes over time. Hence, classical logic can account for nontemporal information about a subject domain. Temporal logics are designed to capture *dynamic change*s over time. There exist three main approaches to the formulation of temporal information using logic: (1) the *method of temporal arguments* (*MTA)* of Haugh (1987) and Bacchus et al. (1991); (2) *modal temporal logics* (Manna & Pnueli, 1981; Reichgelt, 1989); and (3) *reified temporal logics* (McDermott, 1982; Allen, 1984; Kowalski & Sergot, 1986; Dean & McDermott, 1987; Shoham, 1988; Reichgelt, 1989).

In MTA, one simply extends predications and functional terms with extra arguments that stand for the time points or intervals in which the proposition is true or the operation occurs. The main advantage of this method is that it does not break the framework of first-order logic. Consequently, standard theorem-proving techniques can be used for inferencing. Modal temporal logics account for the time dimension by complicating the model theory. The single semantical world of classical logic is replaced by a set of *possible worlds*, where each world is a full semantical model of classical logic, standing for a point in time. Temporal ordering is captured by an accessibility relationship among the possible worlds. Various temporal orderings give rise to a variety of temporal logics. Modal temporal logics extend the classical logic by introducing modal operators for capturing behavior over time. Examples of such operators are "at all future time," "at some future time," "at all past time," and "at some past time."

However, both approaches are limited in their ability to account for temporal common sense because they cannot express general temporal knowledge, such as knowledge about the temporal relationship between events and their effects, or between events and their causes. Reified temporal logics were introduced in AI to enable reasoning about such knowledge. The main characteristic of a reified temporal logic is that propositions are turned into terms that can be named, and to which predicates can be applied. The two most frequent predicates in reified temporal logics are *HOLDS* and *OCCURS*. Both predicates state that a certain named proposition is true at a certain time or holds over a time interval. *HOLDS* applies to *property propositions* like "Mary is happy," while *OCCURS* applies to event propositions, like "Mary met Bob."

We now formulate Example 1 within the reified temporal logic introduced in (Vila & Reichgelt, 1996). The formulation should be understood on an intuitive basis, as a formal introduction of the logic is out of the scope of this paper. Because the example includes quantitative information (numeric time values), it requires also formal axiomatization of continuous time. An example of such a formulation appears in (Davis, 1990; Chapter 4), where a language of real-valued quantities and functions is developed. Davis' formulation enables computations over the real numbers or over subsets of the reals, such as the integers or the rationals. This formulation can be adapted to continuous time be specifying that the space of clock times is isomorphic to the reals. We assume that such a formulation is available, and use numerical values, functions and relations, freely, in the formulation.

EXAMPLE 2. Reformulation of Example 1, using a reified temporal logic.

1. In the first stage, a solder paste is applied to the upper and lower parts within 8 to 10 s:
   OCCURS(*apply*(*solder-paste*,*upper-part*,t_{2s},t_{2e})) **and**
   OCCURS(*apply*(*solder-paste*,*lower-part*,t_{2s},t_{2e})) **and**
   $8 \leq t_{2e} - t_{2s} \leq 10$.
2. In the second stage, screws are fastened to the parts, either with the Pneumatic screw driver within 10 to 17 s, or with the Vertical Mill within 8 to 9 s:
   [*OCCURS*(*fasten*(*screws*,*upper-part*,Pneumatic-screwdriver,t_{3s},t_{3e})) **and**
   OCCURS(*fasten*(*screws*,*lower-part*,Pneumatic-screwdriver,t_{3s},t_{3e})) **and**
   $10 \leq t_{3e} - t_{3s} \leq 17$] **or**
   [*OCCURS*(*fasten*(*screws*, *upper-part*, Vertical-mill, t_{3s},t_{3e})) **and**
   OCCURS(*fasten*(*screws*, *lower-part*, Vertical-mill, t_{3s},t_{3e})) **and**
   $8 \leq t_{3e} - t_{3s} \leq 9$].
3. In the third stage, the ERV7 Robot picks the sealing with the vacuum tool and carries the sealing to the part on the fixture within 12 to 17 s:
   OCCURS(*pick*(*ERV7*,*sealing*,vacuum-tool,t_{4s},t_{4e})) **and**
   OCCURS(*carry*(*ERV7*,*sealing*,upper-part,t_{4s},t_{4e})) **and**
   OCCURS(*carry*(*ERV7*,*sealing*,lower-part,t_{4s},t_{4e})) **and**
   $12 \leq t_{4e} - t_{4s} \leq 17$.
4. The system requires an initial setup within 65 to 85 s:
   $65 \leq t_{1e} - t_{1s} \leq 85$.
5. The conveyor transports the parts from stage to stage within 13 s:
   $t_{2s} - t_{1e} = 13$ **and**
   $t_{3s} - t_{2e} = 13$ **and**
   $t_{4s} - t_{3e} = 13$.
6. The second stage has to be completed within 30 to 40 s after the solder paste is applied:
   $30 \leq t_{3e} - t_{2e} \leq 40$.
7. The third stage has to be completed within 50 to 75 s after the solder paste is applied:
   $50 \leq t_{4e} - t_{2e} \leq 75$.
8. The total assembly time of the product is constrained to be within 120 to 130 s:
   $120 \leq t_{4e} - t_{1s} \leq 130$. ■

Viewing carefully the formulations of Example 1 in a reified temporal logic and in the TCSP model, we see that the metric TCSP network is simply a part of the temporal logic formulation. The TCSP model represents only the temporal relations between the time points and ignores the rest of the knowledge. This concentration turns the problem decidable, and enables the development of specialized inference algorithms that can decide whether the temporal requirements are consistent, test for temporal entailment of relationships between intervals or time points, and enumerate all solutions for the important time points in the process.

We see that the TCSP model is not a competitive model for the temporal logic approach, but simply a restricted account for a portion of the knowledge, that deals only with the temporal elements. Consequently, the TCSP formulation falls short of capturing general periodic knowledge such as:

- The solder past gets dry within 75 s:
  (for all Individual p,
  Time points t_s,t_e:
  $OCCURS(apply(solder\text{-}paste,p,t_s,t_e)) \rightarrow HOLDS(dry(solder\text{-}paste,t_e + 75))$).

As mentioned above, there are efforts to combine the TCSP approach, so as to extend its expressiveness, within the decidability range.

3. QUANTITATIVE TEMPORAL CONSTRAINT SATISFACTION MODELLING

A *quantitative* (*metric) temporal constraint satisfaction problem* (TCSP) is a set of propositions that temporally constrains a collection of binary events. A *binary event* is an ordered pair of *time points*, called *beginning* and *ending*. The propositions constrain the temporal (metric) distance between the points of the events. The task is to suggest a set of events that satisfy the propositions. Each such set comprises a solution. The problem is inconsistent if it has no solution.

3.1. Definitions, concepts and properties

DEFINITION 1. Metric TCSP: Let D be a domain of time points. We take D as the set of real numbers. For $a \le b$, an interval $[a, b]$ over D is the set $\{d | a \le d \le b\}$ in D. If $a = b$, $[a,b]$ is denoted $[a]$. A *TCSP* is a pair $\langle X,\Psi\rangle$, where

1. X is a set of variables $\{X_1,...,X_n\}$, each identified with a distinguished beginning or ending point of an event in the problem domain.
2. Ψ is a set of binary propositions. A binary proposition is a binary constraint, that is, a set of intervals over D that constrains the temporal distance between two variables, to belong to one of the intervals. A proposition that sets the binary constraint C on the variables X_i and X_j is denoted C_{ij}. ■

A TCSP can be graphically described by a directed graph, where vertices represent variables and labeled directed edges represent propositions. That is, the proposition C_{ij} is described by a directed edge $X_i \rightarrow X_j$ that is labeled by the set of intervals C.

EXAMPLE 3. Consider the directed constraint graph of Figure 2, which shows a TCSP with four variables X_1, X_2, X_3, and X_4. The propositions in the problem are: $\{[0],[2]\}_{1,2}$, $\{[-4,-1],[3,6]\}_{1,3}$, $\{[-1,3],[5,6]\}_{1,4}$, $\{[1],[3]\}_{2,4}$, and $\{[2],[4]\}_{3,4}$. ■

DEFINITION 2. Semantics of TCSP: Given a TCSP $\wp = \langle X,\Psi\rangle$.

1. A *proposition* C_{ij} in Ψ is satisfied by an assignment $\{X_1 = t_1,...,X_n = t_n\}$, if $t_j - t_i \in C$.

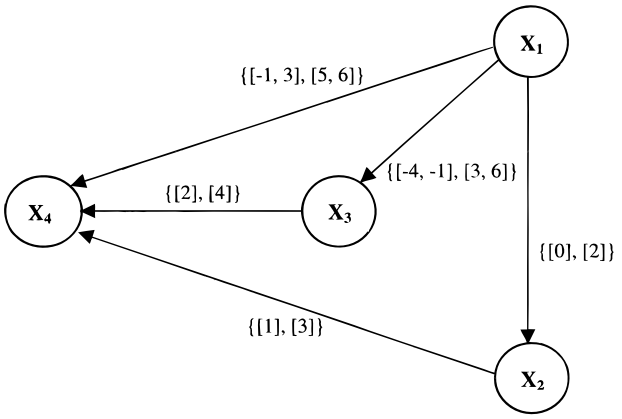

Fig. 2. A directed constraint graph of Example 3.

2. A *solution* is a tuple $(t_1,...,t_n)$ such that the assignment $\{X_1 = t_1,...,X_n = t_n\}$ satisfies every proposition in Ψ. The t_i in a solution $(t_1,...,t_i,...,t_n)$ is a *feasible value* of X_i.
3. A TCSP is *consistent* if it has a solution.
4. A time point t is a *feasible value of a proposition* C_{ij} in Ψ, if there exists a solution such that $t_j - t_i = t$. ■

EXAMPLE 4. The assignment $\{X_1 = 0, X_2 = 2, X_3 = 3, X_4 = 5\}$ satisfies all propositions of Example 3, and 0, 2, 3 and 5 are feasible values of X_1, X_2, X_3, and X_4, respectively. Hence, the tuple (0, 2, 3, 5) is a solution of the problem, and the problem is consistent. The time points 2, 3, 5, 3 and 2 are feasible values of the propositions $\{[0],[2]\}_{1,2}$, $\{[-4,-1],[3,6]\}_{1,3}$, $\{[-1,3],[5,6]\}_{1,4}$, $\{[1],[3]\}_{2,4}$, and $\{[2],[4]\}_{3,4}$, respectively. ■

The intervals in a TCSP are assumed to be nonempty and closed. The term *time slice* is used in an intuitive manner, to stand for a "continuous" portion of the set of time points. Consider the following temporal proposition: "the second operation has to be completed either within 3 to 5 s or within 4 to 6 s or within 6 to 7 s after the paste is applied." The syntactic constraint specified in this proposition is {[3,5],[4,6],[6,7]}, which has complex syntactic representation but a simple semantic denotation; that is, it denotes the single time slice [3,7].

A *temporal constraint* is any set of time intervals; it can be empty, finite or infinite. A proposition with an empty constraint is not satisfiable. A qualitative constraint such as in the proposition "the second operation has to be completed after the paste is applied" is captured by an infinite constraint. A temporal constraint, in general, denotes several (disjoint) time slices. A constraint that denotes a single time slice is called *simple* and a proposition with a simple constraint is a *simple proposition*. A *simple TCSP* is a problem with simple propositions alone. Otherwise it is nonsimple. It is well known (Dechter et al., 1991) that deciding the consistency of nonsimple TCSPs is NP-complete. For a simple

TCSP with n variables, consistency can be decided in $O(n^3)$ time, where n is the number of variables in the problem.

A *simple case* of a TCSP $\wp$ is obtained from $\wp$ by dropping in every proposition of $\wp$ all intervals but one. A *union of problems* is a problem on the same set of variables, whose propositions are the union of the corresponding propositions in the input problems. A nonsimple TCSP can be viewed as the union of its simple cases. Two problems with a common set of variables are *equivalent*, if they have the same set of solutions.

A TCSP is expected to provide the following services:

1. find whether the problem is consistent;
2. find the feasible values of the propositions and of the variables; and
3. find a single solution or all solutions.

The notion of a *minimal problem* of a TCSP, which intuitively is the equivalent reduced TCSP, is introduced in Dechter et al. (1991) as a means for fulfilling the above first two tasks. As for the third service, for a simple minimal problem, a solution can be computed in $O(n^2)$. For a nonsimple minimal problem, finding a solution might be intractable. The minimal problem can be viewed as a compact way for storing all solutions. The definition of a minimal problem requires a means for comparison of TCSPs:

DEFINITION 3. Comparable and Equivalent TCSPs, the Tighter than relation, and Complementation of TCSPs: TCSPs $\wp_1$ and $\wp_2$ are *comparable* if they have the same set of variables, and their propositions are defined on the same pairs of variables. Given two TCSPs $\wp_1$ and $\wp_2$ (not necessarily comparable), $\wp_1$ and $\wp_2$ are *equivalent* ($\wp_1 \equiv \wp_2$), if they have the same set of solutions. Given two comparable TCSPs $\wp_1$ and $\wp_2$, the proposition C_{ij}^1 of $\wp_1$ *is tighter than the proposition* C_{ij}^2 of $\wp_2$ ($C_{ij}^1 \subseteq C_{ij}^2$) if $C^1 \subseteq C^2$. $\wp_1$ *is tighter than* $\wp_2$($\wp_1 \subseteq \wp_2$), if for all $1 \le i, j \le n$, $C_{ij}^1 \subseteq C_{ij}^2$. The *complemented problem* of a TCSP $\wp$, **complement**($\wp$), is a TCSP that is obtained from $\wp$ by complementing all "missing" propositions C_{ij} in $\wp$ (i.e., missing edges in the directed constraint graph) as the *universal propositions* $\{[-\infty, +\infty]\}_{ij}$. ■

DEFINITION 4. Minimal TCSP: The *minimal problem* of a TCSP $\wp$, denoted **min**($\wp$), is the tightest problem among all equivalent and comparable problems of **complement**($\wp$). A problem $\wp$ is *minimal* if $\wp = \mathbf{min}(\wp)$. The minimal problem always exists and is unique by its definition. It exists because for a given TCSP, the set of its equivalent TCSPs is closed under intersection. ■

3.2. Solving TCSPs

In a minimal problem, testing consistency is immediate because it amounts to testing whether any proposition is empty (Dechter et al., 1991). Feasible values of propositions are already given in the propositions. Feasible values for variables can be obtained by first assigning the time point zero to an arbitrary variable X_k. The feasible values of a variable X_i are taken as the allowed values of C_{ki}.

A solution for a simple minimal TCSP can be computed in a backtrack-free manner, by anchoring a variable at a time point, and extending the solution, point by point, such that a new point is always consistent with points already in the solution. This is possible since a simple minimal TCSP has the property that every partial solution can be extended into a full solution. Hence, the solution can be obtained in time $O(n^2)$ (Dechter et al., 1991). If we can find a solution in that way, we say that the problem is *backtrack free* (k-consistent, for every k, in the general CSP terminology). Nonsimple minimal problems are not necessarily backtrack free, as demonstrated in Example 5 below. Consequently, finding a solution might lead to exhaustive search over all simple cases. Yet, the minimal problem might help in filtering out some impossible combinations.

EXAMPLE 5. The following directed constraint graph illustrates a nonsimple minimal problem, which is not backtrack free (Fig. 3). It can be checked directly that every value allowed by any proposition is feasible. Hence, the problem is minimal. Yet, the partial solution (0, 2, 3) assigned to X_1, X_2, and X_3, respectively, cannot be extended to a full solution. Therefore, the problem is not backtrack free. ■

The minimal problem of a TCSP $\wp$ is the tightest problem among all equivalent and comparable problems of **complement**($\wp$). Hence, a problem is minimal if and only if its propositions allow only feasible values. Such propositions are called *minimal propositions*. Theorem 1 below provides a criterion for minimality (Balaban & Rosen, 1999). An equivalent criterion, based on the notion of a distance graph, is provided in (Dechter et al., 1991).

THEOREM 1. *In a simple problem, all propositions are minimal if and only if every proposition C_{ij} is tighter than or equal to all indirect propositions between X_i and X_j.* ■

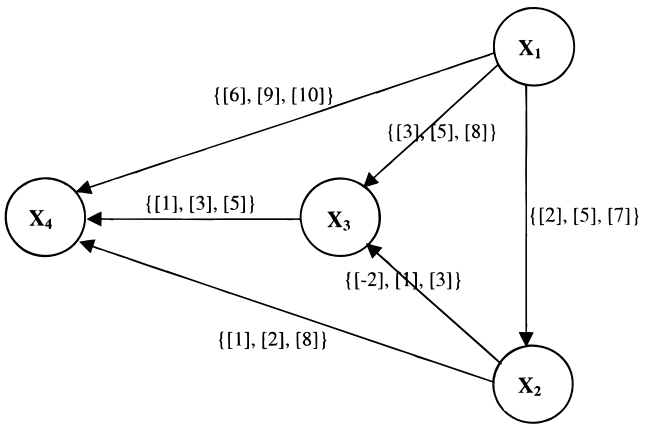

Fig. 3. A directed constraint graph that illustrates a nonsimple minimal TCSP.

In simple problems, because a proposition consists of a single interval, the **All-shortest-path** algorithm of Floyd (Cormen et al., 1990) can be adapted for computing the tightest propositions between all pairs of time points. The **Min-of-Simple** algorithm below exploits this idea to compute the tightest propositions, which by Theorem 1, yield the minimal problem of a given TCSP. The algorithm is taken from (Balaban & Rosen, 1999). A similar algorithm, applied to the *distance graph* of a simple TCSP is given in (Dechter et al., 1991). Because the **Min-of-Simple** process does not divide intervals, the minimal problem of a simple problem is also simple. In a nonsimple problem $\wp$, the minimal problem can be constructed from the minimal problems of the simple cases of $\wp$.

To describe the **Min-of-Simple** algorithm, we need the following binary operations on propositions: *composition* ($\oplus$) and *intersection* ($\cap$). The composition of propositions C_{ij} and C_{jk}, denoted $C_{ij} \oplus C_{jk}$, is defined as follows: For intervals $[a,b],[c,d],[a,b] \oplus [c,d] = [a + c, b + d]$; and $C_{ij} \oplus C_{jk} = \{[a,b] \oplus [c,d] | [a,b] \in C_{ij}, [c,d] \in C_{jk}\}_{ik}$. The intersection of propositions C_{ij} and C_{jk}, denoted $C_{ij} \cap C_{jk}$, is defined as follows: $C_{ij} \cap C_{jk} = \{[a, b] \cap [c,d] | [a,b] \in C_{ij}, [c,d] \in C_{jk}\}_{ik}$. The **Min-of-Simple** algorithm is described as follows:

Algorithm Min-of-Simple
input: A simple TCSP $\wp$
output: min($\wp$)
time: $O(n^3)$, for a problem with n variables.

1. complement $\wp$
2. for every time variable X_i do
3. for every proposition C_{jk}
4. $C_{jk} \leftarrow C_{jk} \cap (C_{ji} \oplus C_{ik})$
5. IF: $C_{jk} = \varnothing$
   THEN: exit (the problem is inconsistent).

4. APPLICATION OF METRIC NONBINARY TEMPORAL CONSTRAINT SATISFACTION TO PROCESS PLANNING

The TCSP framework introduced in Section 3 is limited to problems involving constraints (disjunctive sentences) on pairs of time points. Such constraints, however, are insufficient for a process planning environment, and we must allow nonbinary constraints. In this section, we consider two scenarios of process planning that cannot be modeled by binary TCSPs because they require nonbinary constraints. In Section 4.3 we introduce the n-TCSP model, which is a generalization of the binary TCSP model to account for n-ary metric temporal constraints. We also show that the TCSP model is a degenerate case of the proposed framework, and that all techniques and theoretical results developed for binary TCSPs still hold with the extended model with no extra computational cost.

4.1. PCB Assembly example

Consider the material flow in an automated printed circuit board (PCB) assembling environment, which uses surface-mount technology (Coombs, 1988). Surface mount technology involves mainly the process of placing a surface-mount component onto the board in the required position. Assembly of a surface-mount device involves positioning the component on the board, which has previously had the solder paste applied. Thus, the material handling system has to transport the boards in-between insertion machines (surface-mount devices) and also between insertion machines and buffers at the right time before the adhesive becomes dry (in which case the board has to be discarded).

More specifically, the Flexible Manufacturing System (FMS) under study has two PCB types (PCBs 1 and 2), five workstations (M_0 to M_4) and two material handling systems (a linear conveyor and a SCORBOT-ER7/9 robot). The assembly of both PCBs is performed through three stages of operations (solder paste, passive components assembly, and flat packs assembly) as depicted in Figure 4. Some machines in the system are dedicated to one operation and a single PCB type (M_1, M_3, and M_4); others are flexible and capable of performing multiple operations (M_0 and M_2). The system requires an initial setup within 65 to 85 s. Raw panels of each PCB type are screened via an epoxy compound at the first stage (by M_0), and are transported by the linear conveyor to the surface-mount devices (at the second and third stages), where the components are placed on to the pasted PCBs.[1] The linear conveyor transports the PCBs from the first stage to the second stage, while the PCBs are transported from the second stage to the third stage by the SCORBOT-ER7/9 robot.[2] The robot's material handling task is performed according to a first come first serve (FCFS) scheduling policy. Tables 1 and 2 show the process plans of both PCB types. The transportation times of the material handling systems are provided in Table 3.

Because the applied paste (during stage 1) is usable for a limited amount of time, the second and third operations of both PCB types have to be completed within 30 to 40 s and within 50 to 60 s after the paste is applied, respectively. The total assembly time of both PCB types is constrained to be within 100 to 120 s.

Given the above temporal information, the following queries may be considered: (1) "Is it possible that the assembly process terminates as scheduled (within 100 to 120 s)?"; (2) "Is it possible that in a scenario, which is consistent with the information provided, M_2 is used for the flat packs assembly of PCB type 1 (third stage)?"; (3) "Is it possible that in a scenario, which is consistent with the information provided, the robot transports PCB type 1 from the second stage to the third stage *before* PCB type 2 is served?"; (4) "What

[1] For illustrative purposes, we assume (w.l.o.g.) that PCBs are not delayed between the first and second assembly stages.

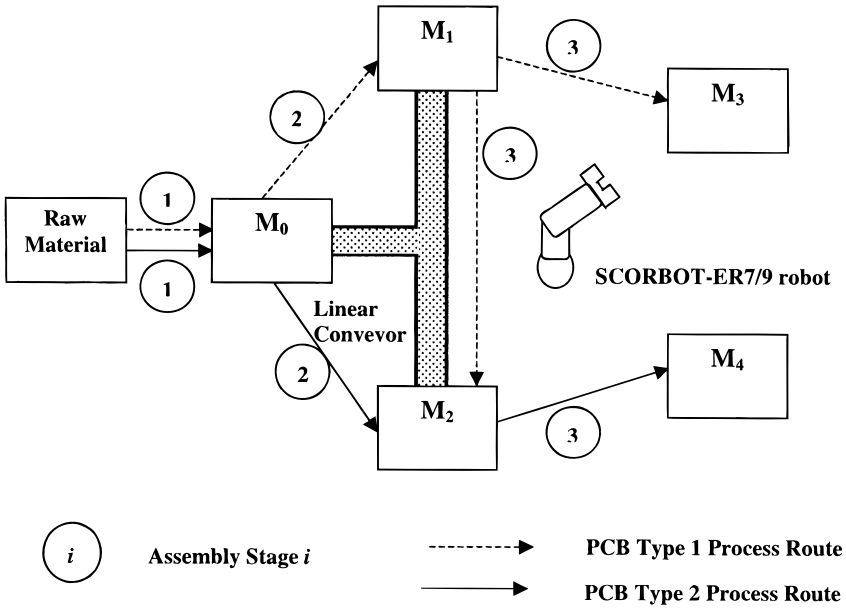

Fig. 4. Schematic of the printed circuit boards process plans.

are the possible times at which the assembly operations can occur?"

Several temporal constraints are given in the process planning description such as (1) "a solder paste is applied to PCB type 1 within 8 to 10 s"; (2) "passive components are assembled to PCB type 2 either with M_2 mode #1 within 25 to 28 s or with M_2 mode #2 within 17 to 25 s"; (3) "flat packs has to be assembled to PCB type 2 within 50 to 60 s after the paste is applied"; (4) "the linear conveyor transports PCB type 1 from the first stage to the second stage within 7 s"; (5) "the total assembly time of PCB type 2 is constrained to be within 100 to 120 s."

Given temporal constraints of this kind, the translation into binary TCSP constraints is straightforward. In contrast, the following additional temporal constraints in the above description cannot be encoded by binary TCSP constraints: (6) "if the passive components are assembled to PCB type 1 *before* the passive components are assembled to PCB type 2 than the SCORBOT-ER7/9 robot transports PCB type 1 from the second stage to the third stage within 5 to 9 s and the flat packs are assembled by M_3 to PCB type 1 within 7 to 12 s; otherwise, if the passive components are assembled to PCB type 1 *after* the passive components are assembled to PCB type 2 and the SCORBOT-ER7/9 robot is available (i.e., PCB type 2 was already transported to the next stage by the robot), than the SCORBOT-ER7/9 robot transports PCB type 1 from the second stage either within 4 to 7 s to M_2 or within 5 to 9 s to M_3. Accordingly, the flat packs are assembled to PCB type 1 within 12 to 17 s by M_2 or within 7 to 12 s by M_3"; (7) "if the passive components are assembled to PCB type 2 *before* the passive components are assembled to PCB type 1, than the SCORBOT-ER7/9 robot transports PCB type 2 from the second stage to the third stage either within 6 to 9 s (using mode #1) or within 3 to 7 s (using mode #2), and the flat packs are assembled to PCB type 2 within 15 to 20 s by M_4; otherwise, if the passive components are assembled to PCB type 2 *after* the passive components are assembled to PCB type 1 and the SCORBOT-ER7/9 robot is available, than the SCORBOT-ER7/9 robot transports PCB type 2 from the sec-

Table 1. *Process plan for PCB type 1*

Operation	Machine candidate	Operating modes	Processing time (s)
Solder paste	M_0	1	[8, 10]
Passive components assembly	M_1	1	[18, 24]
Flat packs assembly	M_2	1	[12, 17]
	M_3	1	[7, 11]

Table 2. *Process plan for PCB type 2*

Operation	Machine candidate	Operating modes	Processing time
Solder paste	M_0	1	[13, 17]
Passive components assembly	M_2	1	[25, 28]
		2	[17, 24]
Flat packs assembly	M_4	1	[15, 20]

Table 3. *Material handling transportation times*

Route direction	Material handling system	Operating modes	Transportation time
$M_0 \rightarrow M_1$	Linear Conveyor	1	[7, 7]
$M_0 \rightarrow M_2$			[10, 10]
$M_1 \rightarrow M_2$	SCORBOT-ER7/9 robot	1	[4, 7]
$M_1 \rightarrow M_3$	SCORBOT-ER7/9 robot	1	[5, 9]
$M_2 \rightarrow M_4$	SCORBOT-ER7/9 robot	1	[6, 9]
		2	[3, 5]

ond stage to the third stage either within 6 to 9 s (using mode #1) or within 3 to 7 s (using mode #2), and the flat packs are assembled to PCB type 2 within 15 to 20 s by M_4."

Table 4. *List of activities and temporal order for a product development process*

Activity	Resource type	Time estimates	Immediate predecessors
A	1	[11, 13]	–
	2	[9, 10]	
B	1	[7, 15]	A
C	1	[5, 15]	A
D	1	[8, 16]	B **or** C
	2	[5, 7]	
E	1	[14, 30]	D
F	1	[6, 18]	D
G	1	[25, 41]	F
H	1	[35, 45]	(E **and** G) **or** F
I	1	[10, 28]	A
J	1	[5, 7]	I

4.2. Product development process example

The problem of selecting an assembly sequence is also considered in the light of recent work in concurrent engineering and product development processes (Nevins & Whitney, 1989). Product development processes involve the coordination and control of a set of complex *activities* to accomplish a project. Activities do not stand alone; a *precedence relationship* determines a sequence for undertaking activities. It specifies that one activity cannot start until a set of preceding activities have been completed. Often, (e.g., when a project has never been done before), time estimates for activities involve uncertainty. To incorporate uncertainty into product development processes, activity times are stated in terms of two reasonable time estimates: (1) the *optimistic* time, which is the shortest time in which the activity can be completed; and (2) the *pessimistic* time, which is the longest estimated time required to perform an activity. Moreover, each such activity can be performed by alternative resource types (e.g., by different contractors). Thus, primitive temporal information can be represented by a set of disjunctive intervals over the beginning and ending time points of each activity. The following example illustrates a product development process. The product development process involves 10 major activities. The temporal order among activities is shown in Table 4. The following additional temporal constraints are considered: (1) the product development process is constrained to be completed within 65 to 70 weeks; and (2) activity F has to be completed within 30 to 35 weeks from the start of the process due to limited availability of resources.

Given such temporal information, we aim at deriving answers to queries and tasks such as: (1) "Is it possible that the entire product development process terminates as scheduled (within 65 to 70 s)?"; (2) "Is it possible that in a scenario, which is consistent with the information provided, the first resource type is used for carrying out activity A"; (3) "Is it possible that in a scenario, which is consistent with the information provided, both activities E and G have to be completed *before* activity H can be started?"; (4) "What are the possible times at which the product development activities can occur?"

To answer the above queries, we let the event A^j denote the jth activity.[2] Several temporal constraints are given in the product development process description such as (1) "activity B is performed within 7 to 15 weeks"; (2) "activity D is performed either with resource type 1 within 8 to 16 s or with resource type 2 within 5 to 7 weeks"; (3) "activity F has to be completed within 30 to 35 weeks *after* the product development process is started"; (4) "the total development time is constrained to be within 65 to 70 weeks."

Given temporal constraints of this kind, the translation into binary TCSP constraints is straightforward. In contrast, consider the following high-order temporal constraints that cannot be encoded by binary TCSP constraints: (5) "activity B or C must precede activity D"; and (6) "activity E and G must precede activity H or activity J must precede activity H."

In summary, we presented two examples, and identified cases where nonbinary temporal constraints arise naturally. The first PCB assembly example demonstrates that n-ary constraints can occur when two (or more) operations both need a nonshareable resource (e.g., material handling system or machine). The second product development process example demonstrates that the temporal ordering of activities may involve multiple activities and therefore complex Boolean logic. In Section 4.4, we show how such temporal constraints can be encoded within an extended n-TCSP model, which is introduced in the following section.

[2] For reference purposes, activities are numbered in accordance with their labels as given in Table 4.

4.3. Nonbinary TCSP (*n*-TCSP)

Having introduced two specific process planning scenarios, we are now prepared to consider problems involving *n*-ary temporal constraints, called *n-TCSP*. In general, we show that a problem involving *n*-ary constraints can be translated into a disjunction of *simple* (*binary) TCSPs* (see Section 3.1). Therefore, the solutions to a given *n*-TCSP are the solutions to all simple binary TCSPs in this disjunction. This decomposition scheme exploits the tractable procedures (e.g., the **Min-of-Simple** algorithm presented in Section 3.2) developed for simple TCSPs. Note that although simple problems are tractable, the decomposition of a disjunctive *n*-TCSP (as well as disjunctive binary TCSP) might include an exponential number of simple problems (exponential in the number of binary edges, i.e., n^2, for a problem with n variables), and hence be intractable.

DEFINITION 5. *n*-TCSP: Let D be a domain of time points. An *n-TCSP* is a pair $\langle X, \Psi \rangle$, where

1. X is a set of time variables $\{X_1, \ldots, X_n\}$, each identified with a distinguished time point that marks the beginning or the ending of an event.
2. Ψ is a set of *disjunctive propositions* $\{\vee C_P\}$, where $P \subseteq X$. In a disjunctive proposition $\vee C_P$, C_P is a *compound proposition* set on P, that is, a conjunction of simple binary propositions (intervals over D) set on time points in P. That is, $C_P = \wedge I_{ij}$, where I_{ij} is an interval over D, that constrains the temporal distance between X_i and X_j. In sum, $\Psi = \wedge(\vee(\wedge I_{ij})X_i, X_j \in P)P \subseteq X$. ■

The *n*-TCSP model is a generalization of the TCSP model since a TCSP is a degenerate *n*-TCSP taken over binary subsets of X alone. That is, in a classical TCSP problem, the set of propositions is:

$$\Psi = \wedge(\vee(\wedge I_{ij})X_i, X_j \in \{X_i, X_j\})\{X_i, X_j\} \subseteq X$$

$$= \wedge(\vee I_{ij})\{X_i, X_j\} \subseteq X. \quad (1)$$

The semantics of *n*-TCSP is the same as that of TCSP. Finding the minimal network is defined the same way. A *simple n-TCSP problem* is a nondisjunctive *n*-TCSP, that is, $\wedge((\wedge I_{ij})X_i, X_j \in P)P \subseteq X$. It can be shown that a simple *n*-TCSP is equivalent to the simple TCSP obtained by intersecting all intervals set on each pair of time points within the various compound propositions (order depends on number of compound propositions—possibly exponential in the number of time variables). We term this simple TCSP the *derived TCSP* of the given simple *n*-TCSP. Consequently, the minimal network of a simple *n*-TCSP can be computed by computing the minimal network of its derived TCSP.

A *simple case* of an *n*-TCSP problem is the simple problem obtained by selecting a single element from each disjunct; that is, $\Psi = \wedge((\wedge I_{ij})X_i, X_j \in P)P \subseteq X$. Finding the minimal network of a disjunctive *n*-TCSP is computed by computing the minimal networks of all simple cases, and combining the results into a new, minimal, disjunctive *n*-TCSP problem.

4.4 Detailed modelling by *n*-TCSP

In this section, we show how the temporal constraints involving in the above examples can be encoded within the *n*-TCSP model.

4.4.1. Modelling the PCB assembly example

To answer the above queries, we consider the following operations (or events):

- O_1—setting up the assembly system;
- O_2^j—applying a solder paste to the raw panels of PCB type j;
- O_3^j—passive components assembly of PCB type j;
- O_4^j—flat packs assembly of PCB type j; and
- R^j—transporting PCB type j from the second stage to the third stage.[3]

These operations are associated with 18 time points (the variables we want to constrain), each of which represents a beginning or an ending point of some operation. In particular, let X_{1s} denote the time where O_1 begins, X_{1e} denote the time where O_1 ends, X_{is}^j denote the time operation O_i^j starts, X_{ie}^j denote the time operation O_i^j ends, X_{Rs}^j denote the time where R^j starts while X_{Re}^j denote the time where R^j ends. In addition, we introduce a special time point, X_0, to represent the beginning of the assembly process. The various temporal constraints provided in the process planning description in Section 4.1 are encoded by *n*-ary constraints as shown in Table 5.

4.4.2. Product development process example

To answer the above queries, we let O_i denote the ith activity (see Table 4).[2] These activities are associated with 10 time points, each of which represents a beginning or an ending point of some operation. In particular, let X_{is} denote the time where O_i begins, X_{ie} denote the time where O_i ends. In addition, we let X_0 represent the beginning of the product development process. A partial list of temporal constraints that are translated into *n*-TCSP is shown in Table 6.

5. SUMMARY

In this paper, we propose a formal approach for the integration of process planing and temporal processing and reason-

[3] As mentioned above, it is assumed that PCBs are not delayed between the first and second assembly stages. This assumption can be removed (i.e., PCBs wait in front of a machine if it is busy) by introducing additional events (with beginning and ending time points) associated with the transportation of PCBs by the conveyor.

Table 5. *Encoding the process planning temporal constraints with Hi-TCSP*

Description	Underlying variables ($P \subseteq X$))	Temporal constraints $\{\vee C_P\}$
"The system's set-up is done at the beginning of the assembly process"	X_0, X_{1s}	$\{[0]$ on $X_0, X_{1s}\}$
"The system requires an initial setup within 65 to 85 s"	X_{1s}, X_{1e}	$\{[65,85]$ on $X_{1s}, X_{1e}\}$
"the solder paste is applied to the raw panels of PCB type 1 *after* the system's set-up is completed"	X^1_{2s}, X_{1e}	$\{[0,\infty]$ on $X^1_{2s}, X_{1e}\}$
"the solder paste is applied to the raw panels of PCB type 2 *after* the system's set-up is completed"	X^2_{2s}, X_{1e}	$\{[0,\infty]$ on $X^2_{2s}, X_{1e}\}$
"either the solder paste is applied to PCB type 1 *before* it is applied to PCB type 1 or vice versa"	$X^1_{2s}, X^1_{2e}, X^2_{2s}, X^2_{2e}$	$\{[0,\infty]$ on $X^1_{2e}, X^2_{2s}\}$ **or** $\{[0,\infty]$ on $X^2_{2e}, X^1_{2s}\}$
"a solder paste is applied to PCB type 1 within 8 to 10 s"	X^1_{2s}, X^1_{2e}	$\{[8,10]$ on $X^1_{2s}, X^2_{1e}\}$
"a solder paste is applied to PCB type 2 within 13 to 17 s"	X^2_{2s}, X^2_{2e}	$\{[13,17]$ on $X^1_{2s}, X^1_{2e}\}$
"the linear conveyor transports PCB type 1 from the first stage to the second stage within 7 s"	X^1_{2e}, X^1_{3s}	$\{[7]$ on $X^1_{2e}, X^1_{3s}\}$
"the linear conveyor transports PCB type 2 from the first stage to the second stage within 10 s"	X^2_{2e}, X^2_{3s}	$\{[7]$ on $X^2_{2e}, X^2_{3s}\}$
"passive components are assembled to PCB type 1 within 18 to 24 s"	X^1_{3s}, X^1_{3e}	$\{[18,24]$ on $X^1_{3s}, X^1_{3e}\}$
"passive components are assembled to PCB type 2 either with M_2 mode #1 within 25 to 28 s or with M_2 mode #2 within 17 to 25 s"	X^2_{3s}, X^2_{3e}	$\{[25,28]$ on $X^2_{3s}, X^2_{3e}\}$ **or** $\{[17,24]$ on $X^2_{3s}, X^2_{3e}\}$
"passive components has to be assembled to PCB type 1 within 30 to 40 s after the paste is applied"	X^1_{2e}, X^1_{3e}	$\{[30,40]$ on $X^1_{2e}, X^1_{3e}\}$
"passive components has to be assembled to PCB type 2 within 30 to 40 s after the paste is applied"	X^2_{2e}, X^2_{3e}	$\{[30,40]$ on $X^2_{2e}, X^2_{3e}\}$
"if the passive components are assembled to PCB type 1 *before* the passive components are assembled to PCB type 2 than the SCORBOT-ER7/9 robot transports PCB type 1 from the second stage to the third stage within 5 to 9 s and the flat packs are assembled by M_3 to PCB type 1 within 7 to 12 s; otherwise, if the passive components are assembled to PCB type 1 *after* the passive components are assembled to PCB type 2 and the SCORBOT-ER7/9 robot is available (i.e., PCB type 2 was already transported to the next stage by the robot), than the SCORBOT-ER7/9 robot transports PCB type 1 from the second stage either within 4 to 7 s to M_2 or within 5 to 9 s to M_3. Accordingly, the flat packs are assembled to PCB type 1 within 12 to 17 s by M_2 or within 7 to 12 s by M_3"	$X^1_{3e}, X^2_{3e}, X^1_{Rs}, X^1_{Re}, X^2_{Re}, X^1_{4s}, X^1_{4e}$	$\{\langle[0,\infty]$ on $X^1_{3e}, X^2_{3e}\rangle$ **and** $\langle[0,\infty]$ on $X^2_{Re}, X^1_{3e}\rangle$ **and** $\langle[4,7]$ on $X^1_{Rs}, X^1_{Re}\rangle$ **and** $\langle[12,17]$ on $X^1_{4s}, X^1_{4e}\rangle\}$ **or** $\{\langle[0,\infty]$ on $X^2_{3e}, X^1_{3e}\rangle$ **and** $\langle[0,\infty]$ on $X^2_{Re}, X^1_{3e}\rangle$ **and** $\langle[5,9]$ on $X^1_{Rs}, X^1_{Re}\rangle$ **and** $\langle[7,12]$ on $X^1_{4s}, X^1_{4e}\rangle\}$
"if the passive components are assembled to PCB type 2 *before* the passive components are assembled to PCB type 1, than the SCORBOT-ER7/9 robot transports PCB type 2 from the second stage to the third stage either within 6 to 9 s (using mode #1) or within 3 to 7 s (using mode #2), and the flat packs are assembled to PCB type 2 within 15 to 20 s by M_4; otherwise, if the passive components are assembled to PCB type 2 *after* the passive components are assembled to PCB type 1 and the SCORBOT-ER7/9 robot is available, than the SCORBOT-ER7/9 robot transports PCB type 2 from the second stage to the third stage either within 6 to 9 s (using mode #1) or within 3 to 7 s (using mode #2), and the flat packs are assembled to PCB type 2 within 15 to 20 s by M_4"	$X^1_{3e}, X^2_{3e}, X^2_{Rs}, X^2_{Re}, X^1_{Re}, X^2_{4s}, X^2_{4e}$	$\{\langle[0,\infty]$ on $X^2_{3e}, X^1_{3e}\rangle$ **and** $\langle[6,9]$ on $X^2_{Rs}, X^2_{Re}\rangle$ **and** $\langle[15,20]$ on $X^2_{4s}, X^2_{4e}\rangle\}$ **or** $\{\langle[0,\infty]$ on $X^2_{3e}, X^1_{3e}\rangle$ **and** $\langle[3,7]$ on $X^2_{Rs}, X^2_{Re}\rangle$ **and** $\langle[15,20]$ on $X^2_{4s}, X^2_{4e}\rangle\}$ **or** $\{\langle[0,\infty]$ on $X^1_{3e}, X^2_{3e}\rangle$ **and** $\langle[0,\infty]$ on $X^1_{Re}, X^2_{3e}\rangle$ **and** $\langle[6,9]$ on $X^2_{Rs}, X^2_{Re}\rangle$ **and** $\langle[15,20]$ on $X^2_{4s}, X^2_{4e}\rangle\}$ **or** $\{\langle[0,\infty]$ on $X^1_{3e}, X^2_{3e}\rangle$ **and** $\langle[0,\infty]$ on $X^1_{Re}, X^2_{3e}\rangle$ **and** $\langle[3,7]$ on $X^2_{Rs}, X^2_{Re}\rangle$ **and** $\langle[15,20]$ on $X^2_{4s}, X^2_{4e}\rangle\}$
"flat packs has to be assembled to PCB type 1 within 50 to 60 s after the paste is applied"	X^1_{2e}, X^1_{4e}	$\{\langle[50,60]$ on $X^1_{2e}, X^1_{4e}\rangle\}$
"flat packs has to be assembled to PCB type 2 within 50 to 60 s after the paste is applied"	X^2_{2e}, X^2_{4e}	$\langle[50,60]$ on $X^2_{2e}, X^2_{4e}\rangle$
"the total assembly time of PCB type 1 is constrained to be within 100 to 120 s"	X_0, X^1_{4e}	$\langle[100,120]$ on $X_0, X^1_{4e}\rangle$
"the total assembly time of PCB type 2 is constrained to be within 100 to 120 s"	X_0, X^2_{4e}	$\langle[100,120]$ on $X_0, X^2_{4e}\rangle$

Table 6. *Encoding the product development temporal constraints with Hi-TCSP*

Description	Underlying variables ($P \subseteq X$)	Temporal constraints $\{\vee C_P\}$
"activity O_1 must precede activity O_4"	X_{1e}, X_{4s}	$\{\langle[0,\infty] \text{ on } X_{1e}, X_{4s}\rangle\}$
"activity O_4 is performed either with resource type 1 within 8 to 16 s or with resource type 2 within 5 to 7 weeks"	X_{4s}, X_{4e}	$\{\langle[8,16] \text{ on } X_{1e}, X_{4s}\rangle\}$ **or** $\{\langle[5,7] \text{ on } X_{1e}, X_{4s}\rangle\}$
"either activity O_5 and O_7 must precede activity O_8, or activity O_6 must precede activity O_8"	$X_{5e}, X_{7e}, X_{6e}, X_{8s}$	$\{\langle[0,\infty] \text{ on } X_{5e}, X_{8s}\rangle$ **and** $\langle[0,\infty] \text{ on } X_{7e}, X_{8s}\}$ **or** $\{\langle[0,\infty] \text{ on } X_{6e}, X_{8s}\rangle\}$
"activity O_6 has to be completed within 30 to 35 weeks *after* the product development process is started"	X_0, X_{6e}	$\{\langle[30,35] \text{ on } X_0, X_{6e}\rangle\}$
"the total development time is constrained to be within 65 to 70 weeks"	$X_0, X_{ie} \forall i$	**and** $\{\langle[65,70] \text{ on } X_0,\ X_{ie}\rangle\}$ $1 \le i \le 10$

ing. Our approach relies on constraint-network formalisms, and TCSPs in particular. The TCSP model is weaker than a full temporal logic. We show that a TCSP network is a subset of a formulation using a reified temporal logic, and discuss the advantages of using such a restricted model. In the proposed approach, operations (or activities) are considered as events, each of which is assigned a binary temporal constraint related to the time difference between the beginning and ending time points of the event. Alternative processes for each operation are accounted for by disjunctive temporal binary constraints. Such constraints, however, are shown to be insufficient for our main application of process planning; we must allow nonbinary temporal constraints. It is demonstrated that *n*-ary temporal expressions could arise from a resource constraint or temporal ordering of activities that involves complex Boolean logic. Because binary TCSPs do not offer a convenient mechanism for dealing with such information, we present an extended approach, called *n*-TCSP, that explores TCSPs with nonbinary constraints. We also show that our approach comprises binary TCSP as a special case; and all techniques and theoretical results developed for TCSP apply to the extended model with no computational cost. The proposed approach can be used as preprocessing of selecting an optimal assembly sequence by finding the global optimum among all feasible sequences generated by *n*-TCSP.

To implement our approach, the following aspects of information processing must be well addressed: (1) improving and automating the transformation of a given high-level process plan description (as provided in Sections 4.1 and 4.2) to temporal constraints; (2) maintaining and organizing the temporal knowledge base; (3) developing a retrieval mechanism for queries; and (4) developing an inference mechanism for deriving new temporal information. In addition, a smooth coordination between temporal reasoning and CAD/CAM, as well as other aspects of integrated manufacturing systems is essential to the success of our approach.

A classification of process planning situations involving nonbinary temporal constraints, with the aim of exploiting their algebraic properties, is desirable. Finally, a natural extension of our work is to explore *n*-TCSPs that combine qualitative and quantitative temporal knowledge as pertinent to process planning [as suggested, for example, by Meiri (1996)], and to further extend the expressiveness of this framework with if–then rules [as suggested in the Token-Datalog language of Schwalb et al. (1997)].

REFERENCES

Allen, J.F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM 26 (11)*, 832–843.

Allen, J.F. (1984). Towards a general theory of action and time. *Artificial Intelligence 23(2)*, 123–154.

Bacchus, F., Tenenberg, A.J., & Koomen, J.A. (1991). A non-reified temporal logic. *Artificial Intelligence 52(1)*, 87–108.

Baker, A.B. (1991). Non-monotonic reasoning in the framework of situation calculus. *Artificial Intelligence 49(1–3)*, 5–23.

Balaban, M., & Rosen, T. (1998). STCSP—Structured temporal constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence* (in press).

Chang, T.C. & Wysk, R.A. (1985). *An introduction to automated process planning systems*. Prentice Hall, Englewood Cliffs, New Jersey.

Chomicki, J. (1996). Finite-representation of infinite query answers. *Transactions on Database Systems 2(1)*, 23–34.

Coombs C.F. (1988). *Printed circuits handbook*. McGraw-Hill, New York.

Cormen, T.H., Leiserson, C.E., & Rivest, R.L. (1990). *Introduction to algorithms*. MIT Press, Cambridge.

Davis, E. (1990). *Representations of commonsense knowledge*. Morgan Kaufmann Publishers, San Mateo, California.

Dean, T.L. (1989). Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the ACM 36(4)*, 687–718.

Dean, T.L., & McDermott, D.V. (1987). Temporal data base management. *Artificial Intelligence 32(1)*, 1–55.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence 49(1–3)*, 61–95.

Dechter, R., & Pearl, J. (1987). Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence 34(1)*, 1–38.

DeFazio, T.L., & Whitney, D.E. (1987). Simplified generation of all mechanical assembly sequences. *IEEE Transactions on Robotics and Automation RA-3 (3)*, 640–658.

Freuder, E.C. (1978). Synthesizing constraint expressions. *Communications of the ACM 21(11)*, 958–965.

Freuder, E.C. (1982). A sufficient condition of backtrack-free search. *Journal of the ACM 29(1)*, 24–32.

Golumbic, M.C., & Shamir, R. (1993). Complexity and algorithms for rea-

soning about time: A graph theoretic approach. *Journal of the ACM 40(10)*, 1108–1193.
Haralick, R.M., & Elliott, G.L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence 14(2)*, 263–313.
Harel, D. (1979). First-order dynamic logic. In *Lecture Notes in Computer Science*, (Goos & Hartmanis, Eds.), Vol. 68, Springer Verlag, New York.
Haugh, B. (1987). Non-standard semantics for the method of temporal arguments. *Proceedings of IJCAI-87*, 449–455.
Kautz, H. & Ladkin, P. (1991). Integrating metric and qualitative temporal reasoning. *Proceedings of AAAI-91*, 241–246.
Kowalski, R.A., & Sergot, M.J. (1986). A logic-based calculus of events. *New Generation Computing 4(1)*, 67–95.
Ladkin, P.B., & Reinefeld, A. (1992). Effective solution of qualitative interval constraint problems. *Artificial Intelligence 57(1)*, 105–124.
Lin, A.C., & Chang, T.C. (1991). Automated assembly planning for 3-dimensional mechanical products. *Proceedings of NSF Design and Manufacturing Systems Conference*, 523–531.
Manna, Z., & Pnueli, A. (1981). Verification of concurrent programs: The temporal framework. In *The Correctness Problem in Computer Science*, (Boyer, R.S., & Moore, J.S., Eds.), pp. 215–273. Academic Press, New York.
McCarthy, J., & Hays, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, (Meltzer, B., & Michie, D., Eds.), Vol. 4, pp. 463–502. Edinburgh University Press, Edinburgh.
McDermott, D. (1982). A temporal logic for reasoning about processes and plans. *Cognitive Science 6(2)*, 101–155.
Mackworth, A.K. (1977). Consistency in networks of relations. *Artificial Intelligence 8(1)*, 99–118.
Malik, J., & Binford T.O. (1983). Reasoning in time and space. *Proceedings of IJCAI-83*, 343–345.
Meiri, I. (1996). Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence 87(1–2)*, 343–385.
Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Science 7(1)*, 95–132.
Nevins, J.L., & Whitney, D.E. (1989). *Concurrent design of products and processes*. McGraw-Hill, New York.
Nudel, B. (1983). Consistent-labeling problems and their algorithms: Expected-complexities and theory-based heuristics. *Artificial Intelligence 21(2)*, 135–178.
Pylyshyn, Z. (1987). *The frame problem and other problems of holism in artificial intelligence*. Ablex Publishing, Norwood, New Jersey.
Pratt, V. (1976). Semantical considerations on Floyd–Hoare Logic. *Proceedings of the 17th FOCS*, pp. 109–121. IEEE, Piscataway, New Jersey.
Reichgelt, H. (1989). A comparison of first order and modal treatments of time. In *Logic Based Knowledge Representation* (Jackson, P., Reichgelt, H., & van Harmelen, F., Eds.), pp. 137–168. MIT Press, Cambridge, MA.
Schmidt, L., & Jackman, J. (1995). Evaluating assembly sequences for automatic assembly systems. *IEEE Transactions 27(1)*, 23–31.
Schwalb, E., & Dechter, R. (1997). Processing disjunctions in temporal constraint networks. *Artificial Intelligence 93(1–2)*, 29–41.
Schwalb, E., Vila, L., & Dechter, R. (1997). Decidability and finite representability of deductive databases with temporal constraints. (Technical Report).
Shoham, Y. (1988). *Reasoning about change: Time and causation from the standpoint of artificial intelligence*. MIT Press, Cambridge, Massachusetts.
Valdes-Perez, R.E. (1986). Spatio-temporal reasoning and linear inequalities. *Artificial Intelligence Laboratory, AIM-875*, MIT, Cambridge, MA.
VanBeek, P. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence 58(1–3)*, 297–326.
Vila, L., & Reichgelt, H. (1996). The token reification approach to temporal reasoning. *Artificial Intelligence 83(1)*, 59–74.
Vilain M., & Kautz, H. (1986). Constraint propagation algorithms for temporal reasoning. *Proceedings of AAAI-86*, 377–382.
Woo T.C., & Dutta, D. (1991). Automatic disassembly and total ordering in three dimensions. *Transactions of ASME Journal Engineering for Industry 113(1)*, pp. 207–213.

Mira Balaban received a B.Sc. in mathematics and statistics in 1970 from Tel-Aviv University (Israel), and a M.Sc. and a Ph.D. in computer science from the Weizmann Institute of Science in 1975 and 1982, respectively. She also graduated in music from the Rubin Academy of Music in Tel-Aviv in 1969. Between 1982 to 1988 she held the position of an Assistant professor at The University at Albany—SUNY, NY. From 1988–1995 she held the position of a Senior Lecturer in the Department of Mathematics and Computer Science at Ben-Gurion University of the Negev in Israel. Since 1996 she is associated with the Information Systems Program in the Department of Industrial Engineering and Management in Ben-Gurion University, and with the Department of Musicology at Bar-Ilan University. Her research is in the areas of artificial intelligence, conceptual modelling, and computer music. In artificial intelligence she works on knowledge representation and temporal reasoning, hierarchical planning with time structures, temporal constraint satisfaction, and description logics. In conceptual modelling she works on extensions of the entity relationship model, and in computer music she develops computational tools for music research and processing. Dr. Balaban has published extensively in these areas, and is a co-editor of a paper collection on computer music. She organized several international workshops on AI and music, and served as a member of the program committee of several conferences on computer music and on artificial intelligence.

Dan Braha received a Ph.D. in engineering sciences from Tel-Aviv University, Israel. He served as a Research Associate in the Department of Manufacturing Engineering at Boston University. Currently, he is an Assistant Professor in the Department of Industrial Engineering at Ben-Gurion University, Israel. His research within engineering design focuses on developing methods to help the designer move from the conceptual phase to the realization of the physical device. To achieve this objective, he has developed a mathematical theory—the Formal Design Theory (FDT). His research interests include knowledge-based systems, case-based reasoning, cognitive science, artificial intelligence, computational complexity, information theory, and operations research. Dr. Braha has published extensively in these areas, including a book on the foundations of engineering design.