Using Temporal Constraint Networks for Smart Grid Scheduling

Wouter Smit - 4401409
w.j.smit-1@student.tudelft.nl
Delft University of Technology

***Abstract*—Smart Grid scheduling problems are characterized by quickly changing situations and multiple external factors that cannot be controlled. Most smart grid research applies stochastic models over the total power consumption of a household or system to find a schedule that achieves an optimization, a balancing, or constraint satisfaction. While these solutions are able to efficiently capture the unpredictability in the problem, they only make use of a limited amount of information. These models are not yet able to make flawless decisions, so we try to optimize this by using a model that is able to use more information. A different scheduling model that is able to do so is the Temporal Constraint Satisfaction Problem (TCSP). However, the information that this model needs requires substantial effort to obtain in smart grid environments, especially for academic purposes. For this reason, research into the benefits of doing so is scarce.**

In this paper we attempt to use this model with detailed environmental information, specifically the activation of electric devices in households, to better optimize a motivating smart grid scheduling problem. We find that problems with characteristics typical in smart grids are difficult to express as TCSP. We discuss these characteristics and provide the concept of *non-binary conditions* and *conditional preference* to solve these difficulties and provide more expressive conditional reasoning about optimality in TCSP.

1. Introduction

Smart grid research consists of managing the electricity grid to retain a balance in the production and consumption of electricity. To this end, most smart grid problems deal with finding a schedule that achieves such a balance, based on temporal information about the electricity grid.

The scheduling problems in smart grids are typically complicated by large fluctuations and sudden changes in the environment, as well as the unpredictable (human) behaviour of multiple stakeholders in the electricity grid.

Most research on such scheduling problems uses probabilistic models to decide a planning that is able to foresee and adapt to these situations. This approach is able to model the environment in terms of stochastic distributions, therefore reducing the complexity of all its components. This results in only using global, high-level information, such as total consumption and production patterns. While this is effective, a natural question that arises is whether more detailed information about the environment can be utilized to achieve a better optimization.

A model that is able to capture and reason about detailed temporal information to produce a schedule is the *Temporal Constraint Satisfaction Problem* (TCSP), often referred to as *Temporal Constraint Satisfaction Network* (TCSN), because of its graphical representation. The model is commonly used in AI task scheduling, because it can guarantee correctness (safety) of a schedule, even with complex constraints or on-the-fly changes to the schedule or its constraints.

In this paper we look into the usefulness of temporal networks for smart grid applications, by modelling a motivating smart grid scheduling problem as a temporal network. We discuss the various extensions and what is needed to sufficiently capture the information about the scheduling problem.

In section 2, we provide an overview of the various temporal problem classes and discuss the expressive powers they each provide. In section 3, we describe the motivating smart grid scheduling problem, and we show how its various aspects can be modelled in section 4. Section 5 discusses the benefits and weaknesses of the different approaches that we can take, and section 6 shows what extensions can be defined to improve the model.

2. Background

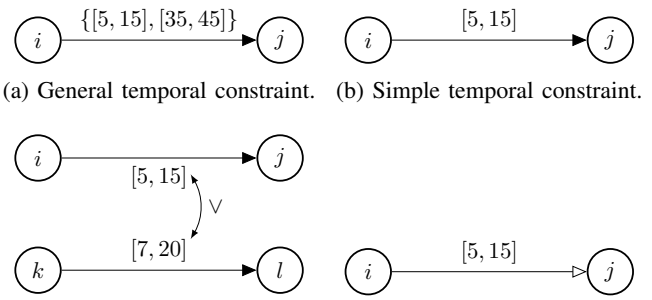
The *Temporal Constraint Satisfaction Problem* (TCSP) [1] is a popular model for verifying constraints in a temporal planning system. It is a pair $\langle \mathcal{T}, \mathcal{C} \rangle$, where $\mathcal{T}$ is a set of variables called *time-points* that denote an instantaneous event occurring at a certain time, and $\mathcal{C}$ is a set of binary *constraints*, that can be described as a set of intervals that bounds the difference between two time-points. The model can be described and solved as a graph, where $\mathcal{T}$ are vertices and $\mathcal{C}$ are edges, in which case it is often called a Temporal Constraint Network (TCN), and the two terminologies are used interchangeably. If an assignment exists to all time-points in $\mathcal{T}$ that respects all constraints in $\mathcal{C}$, the network is considered *consistent*. The model's strength lies herein, that any solution is guaranteed to satisfy all constraints. In practical application, this means that the model can be used to verify the feasibility of a planning.

2.1. Types of Temporal Constraints

The constraints in TCSP can be described as a set of intervals, such that the difference between two time-points X_j and X_i is within one of these intervals, see Figure 1a.

$$\{I_1, ..., I_n\} = \{[a_1, b_1], \ldots, [a_n, b_n]\}$$
$$(a_i \leq X_j - X_i \leq b_1) \vee \cdots \vee (a_n \leq X_j - X_i \leq b_n)$$

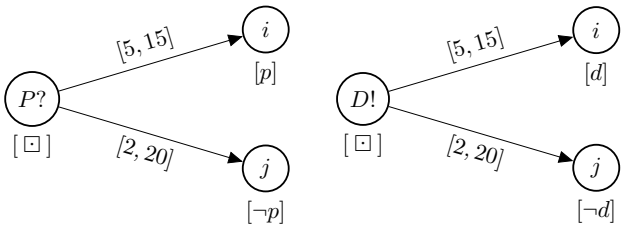
The Simple Temporal Problem (STP / STN) [1] is a well-studied subclass that allows for at most one constraint interval between two nodes, see Figure 1b. Such an interval is called



(a) General temporal constraint. (b) Simple temporal constraint.

(c) Disjunctive temporal constraint between multiple nodes. (d) A contingent (simple constraint) link. j is assigned by Nature.

Figure 1: Temporal constraint types.



(a) An observation node $P?$. (b) A decision node $D!$.

Figure 2: Two concepts of conditionality

simple. Checking the consistency of an STN can be done in polynomial time [1], in contrast to that of a TCN, which is known to be NP-Complete [1]. We give the formal definition of an STP in Definition 1.

Definition 1 (STP [1])**.** A *Simple Temporal Problem* (STP) is a pair $\langle \mathcal{T}, \mathcal{C} \rangle$, where $\mathcal{T}$ is a set of variables in $\mathbb{R}$ called *time-points* and $\mathcal{C}$ is a set of binary *constraints*, specifying a single interval $[a, b]$ between two variables $Y, X \in \mathcal{T}$, so that $a \leq Y - X \leq b$. An STP is *consistent* if an assignment to all variables in $\mathcal{T}$ exists that satisfies all constraints in $\mathcal{C}$.

A third type of constraint is the *disjunctive* constraint. These constraints are not binary, but n-ary. They effectively consist of disjunctions of simple constraints, allowing the network to express such things as 'either X_i and X_j are constrained, or X_k and X_l are'. The example in Figure 1c shows such a constraint for two edges. In general, a disjunctive constraint is described as follows.

$$(a_1 \leq X_{j_1} - X_{i_1} \leq b_1) \vee \cdots \vee (a_n \leq X_{j_n} - X_{i_n} \leq b_n)$$

TCSPs containing such constraints are called *Disjunctive Temporal Problems* (DTP). DTNs are consistent if at least one disjunct can be satisfied for each constraint. They are NP-Hard, but have practical applications.

The STP class has been extended in various orthogonal ways to enhance its expressiveness, because many practical problems deal with more complex situations than a definitive set of events and completely controllable relations. This has led to various properties, which will be discussed below. Similar extensions have been defined for DTP and the general TCSP, which we do not discuss in detail.

2.2. Conditionality

Not always should all events be executed. In some situations, the choice for executing some events depends on an external factor, called an *observation*. The observation is a boolean proposition that describes some state, and is only observed at some time point, described by an *observation node*. For instance, there might be two routes to some destination, but the shorter route might be blocked. This cannot be known before departure, but only observed when arriving at the junction of the two routes. The event for taking the shorter route should only be executed if it is not blocked, and vice versa. Solving the network results in an assignment of time values, but only to the set of events that are executed. Observation nodes are denoted by a question mark, as seen in Figure 2a. In this example, i and its corresponding constraint are only considered if p holds, while instead j and its corresponding constraint are considered when $\neg p$ holds. A '$\boxdot$' symbol is used to denote an empty label, meaning that the variable is always executed. In this text we omit the label entirely if it is empty.

Definition 2 (Observation node)**.** An *observation node* is a variable in $\mathcal{T}$ that, at the moment of its execution, uncovers the truth value of some proposition p. The value of p is decided by an external factor (Nature).

Definition 3 (Label)**.** A label is a conjunction of literals (including negations) that is attached to a variable $v \in \mathcal{T}$. v should only be executed if the label's value becomes *true*.

Definition 4 (CSTP)**.** A *Conditional Simple Temporal Problem* is a tuple $\langle \mathcal{T}, \mathcal{C}, \mathcal{P}, L, O \rangle$, where $\mathcal{T}$ and $\mathcal{C}$ are defined as in an STP, but $\mathcal{T}$ contains a subset of observation nodes $\mathcal{T}_o \subseteq \mathcal{T}$. Furthermore, $\mathcal{P}$ is a finite set of propositions and L and O are functions.
$L : V \rightarrow P^*$ is a function assigning a label to each node in $\mathcal{T}$.
$O : P \rightarrow \mathcal{T}_o$ is a bijective function associating a proposition with an observation node, so that $O(A)$ is the node that provides the truth-value for proposition A.

When conditionality is introduced, the concept of consistency changes slightly. In STP, consistency means that a valid assignment to all time-points exists, but this is not the case in CSTP. The Agent only executes those variables that have a valid label. Instead, a network is consistent if, given a certain knowledge about the uncontrollable conditions, a valid assignment exists to all remaining variables, leading to three notions of consistency: Strong, Weak, and Dynamic consistency. [2]

2.2.1) Strong consistency (SC): When a single assignment exists that makes the network consistent for all possible outcomes of the observations, the network is strongly consistent. In other words, the network can be solved with one schedule

without knowing anything beforehand. This is a very strong property, and few networks are strongly consistent in realistic settings.

2.2.2) Weak consistency (WC): When a valid assignment exists for every possible situation, but the assignment is different per situation, the network is weakly consistent. In other words, if the uncertain parts of the network are fully known beforehand, a valid schedule can be determined. This property has unpractical requirements, because the point of uncertainty is generally that these values are not known.

2.2.3) Dynamic consistency (DC): Since neither hard nor weak consistency captures the ability to find a solution in a practical environment, a third type of consistency is considered. A network is dynamically consistent if, during the execution of the schedule, at each point t in time there exists a valid schedule for the remaining network. The agent is able to use the history of all events and observations happening before t, but has no knowledge over the future events. This is common in realistic settings, because the Agent is usually able to adapt its behaviour during execution.

2.3. Decisions

Following conditionality property, the model has also been extended with decisions, a property that is also known as *controllable conditionality*. Whereas the observations are uncovered by the Agent outside of its control, controllable observations, or *decision nodes*, let the Agent itself decide a value for propositions. They are represented by an exclamation mark, as seen in Figure 2b.

The addition of such observations and decisions, and execution of events based on these conditions, combines the scheduling aspects of STPs with planning. This is a particularly strong extension, because the executability of the schedule is considered while making planning choices.

Definition 5 (Decision node)**.** A *decision node* is a variable in $\mathcal{T}$ that, at the moment of its execution, sets the truth value for some proposition p. In contrast to observation nodes, the value of p is decided by the Agent.

2.4. Uncertainty

While the STP assumes that every event's time value is freely assignable, this is often not the case in reality. An example of this is commuting time: while the departure time can be decided by the commuter, the exact arrival time depends on external factors (Nature), such as the amount of traffic on the journey. The arrival time is considered a *contingent* event. Its time value will be decided by a *contingent* relation with some other non-contingent event (such as the departure time). Contingent links are graphically represented by open arrow heads, as shown in Figure 1d. Their definition is given in Definition 6. The concept of strong, weak and dynamic consistency is renamed to strong, weak and dynamic *controllability* when the network contains uncertainty.

Definition 6 (Contingent link)**.** A *contingent link* is an uncontrollable constraint of the form (A, ℓ, u, C), where $A, C \in \mathcal{T}$, but C is not freely assignable. $[\ell, u]$ denotes the interval that the constraint is bounded by, such that $C - A \in [\ell, u]$ and $\ell > 0$.

Definition 7 (STPU [2])**.** A *Simple Temporal Problem with Uncertainty* is a tuple $\langle \mathcal{T}, \mathcal{C}, \mathcal{L} \rangle$ where $\mathcal{T}$ and $\mathcal{C}$ are defined as in an STP, $\mathcal{T}$ is further partitioned into two disjunctive sets T_a and T_c representing assignable variables and contingent variables. $\mathcal{L}$ is a set of contingent links, as defined in Definition 6.

2.5. Preferences

The constraints in TCSP are very strict: they may not be violated even slightly. However, many temporal relations are not so strictly defined. While the exact time difference between two events has a desired value, deviating slightly from this ideal value is not impossible, and can be allowed. An example of such a situation is the constraint of taking medication ten minutes after exercise. While ten minutes is optimal, nine or eleven minutes is not much worse.

To express such softer constraints, and preference of certain time points over others, a preference function is introduced, that maps the range of time values in the constraint interval to a preference value. Solving the STP is then not only concerned with finding a valid assignment, but also with achieving the highest possible total preference value.

Definition 8 (Soft constraint)**.** A *soft constraint* is a temporal constraint $\langle I, f \rangle$, where $I = [l, u]$ is an interval, restricting the difference between two time-points as in the classical constraint definition, and $f : I \to A$ is a *preference function*, mapping each value in I to a preference value $a \in A$. A has a total order that is imposed by a semiring.

2.6. Existing Extensions

The above properties have been captured in various class definitions, that combine one or multiple properties. Literature has taken orthogonal paths with conditional and non-conditional variants of the temporal problem. Defining classes is not always a simple matter of adding the specific relations or nodes to the model, although it can be, because the interplay between them sometimes conflicts. See Table I for a summary of the currently defined classes that implement these four properties.

2.7. Solvability and practical application

Although we do not go into detail about all aspects of the solvability of the classes and the complexity thereof, it is relevant to know the general complexity of the various classes. The complexities are given in Table I as well. For the classes that differentiate between SC, WC and DC, we talk about dynamic consistency only, because this is the most valuable and practically useful property. Also, this is the property that we require in our motivating problem in section 3.

The earlier branch of classes, that lack conditionality, have been shown to be solvable in polynomial time. The relatively

Name	**C**onditionality	**U**ncertainty	**D**ecisions	**P**references	Complexity
TCSP [1]	-	-	-	-	NP-Hard [1]
DTP [1]	-	-	-	-	NP-Hard [1] [3]
STP/STN [1]	-	-	-	-	P [1]
STPP [4]	-	-	-	Yes	P [4]
STPU [2]	-	Yes	-	-	P [5] [6]
STPPU [7]	-	Yes	-	Yes	P [7]
CSTP [8]	Yes	-	-	-	EXP [9] [10]
CSTPP [11]	Yes	-	-	Yes	NP-Hard [12]
CSTNU [13]	Yes	Yes	-	-	EXP-Complete [14]
CSTND [15]	Yes	-	Yes	-	EXP-Complete [14]
CSTNUD [16]	Yes	Yes	Yes	-	EXP-Complete [14]

Table I: Defined problem classes and their properties.

newer branch of extensions is based on CTP (and CSTP for its analogue with STP) [8]. In two orthogonal directions, CTP has been extended with preferences, and with uncertainty. The latter has been further extended with decisions. While practical algorithms exist for CSTP and CSTPP, there are currently no reasonably tractable solutions for CSTNU, CSTND, and CSTNUD. There exist a sound, but not complete propagation-based algorithm based on [10] for CSTNU [13]. These classes have also been translated into a Timed-Game Automaton (TGA) [17], where dynamic consistency/controllability is translated to a state reachability problem. While this is the only sound and complete solution for these classes, this problem is extremely difficult as well.

3. Motivating Problem

The motivating problem is an optimization of a photovoltaic (PV) system. In shared-roof housing, one such system is used to provide multiple households with solar energy. Because a PV system can generally only supply one household at a time, the generated electricity must be distributed somehow by time-sharing the PV system. One way to do this, is to provide every household with an (approximately) equal share of solar energy, calculated over the course of a year.

While this approach is reasonable, it does not encompass the optimality of *self-consumption*. Self-consumption is the ratio of produced solar energy that is instantaneously consumed by the connected household, thereby preventing feed-in into the electricity grid. A higher rate of self-consumption is beneficial to the household, because it is cheaper to use the electricity than to sell it to the energy provider, as well as to the network operator, because the reduced feed-in causes less strain on the peak capacity of the grid.

$$\text{self-consumption} = \frac{min(E_{used}, E_{generated})}{E_{generated}}$$

In a single-household setting, the only way to manipulate the self-consumption is by adapting the consumption pattern of the household to the PV production, or by using batteries to store excess solar energy during peaks in production for later consumption. In a multi-household setting, we have the extra option of switching the PV system to another household, that is consuming more energy and could therefore utilize more of the solar energy. The times-sharing schedule for the PV system can exploit this to optimize the self-consumption without any investment costs or adapted consumption behaviour.

The motivating problem thus is to find a schedule that equally shares the produced solar energy among the households, while achieving the highest possible self-consumption.

To aid the optimization, monitoring devices have been installed on major energy consumers in each household, such as whitegood appliances like washers and dryers, and heatpumps. These devices can detect when a device turns on or off, and measure the power drawn during an active cycle of each appliance. Furthermore, these measurements are aggregated in a central system that has a live communication with the PV system and can signal a command to switch its connection in real time.

Definition 9 (Problem instance)**.** The problem input is defined by the tuple $\langle D, C, E \rangle$, where D is the set of devices, C is the set of households, called channels. Each device belongs to exactly one channel as defined by $C(d_i) = c_j$ where $d_i \in D$ and $c_j \in C$. E is the set of activation and deactivation events, generically called *device events*. We describe the events for device $i \in D$ at some time t as $on_{i,t}$ and $off_{i,t}$ respectively, and omit the timestamp in the remaining text when it is unambiguous. We call the currently connected channel the *active* channel. The active channel (c_{act}) at some time t is described by the function $act(t)$. We call the moment at which the active channel changes a *switch*, and again implicitly denote a switch at time t by s_t. Because a slight cost is incurred for every switch, the system should only switch if a significant increase can be achieved.

The solution is a schedule of switches, each describing a time and a channel.

4. Modelling

We now discuss various approaches to modelling the problem. We present two network models, and multiple ways to reason about the optimality of a switch decision. We postpone the comparison of the different options to section 5.

It becomes clear that we need variables (nodes) for the device events and the switching events, since these can be represented as instantaneous points in time. We also define a start node X_0, representing absolute time.

4.1. Structure of the problem

The problem scenario changes only when a device event occurs. Such an event can therefore lead to a switch. This results in a sequence of two steps that is repeated.

1) A device event happens.
2) The scheduler decides to switch

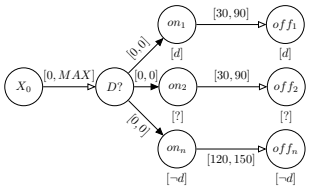
Because of a finite space and the online nature of the problem, we use a sliding window to look ahead a limited amount of time. We define the window size as $[0, MAX]$, making a constraint with this interval effectively unconstrained. We now describe various approaches to modelling these two steps.

4.2. Modelling device events

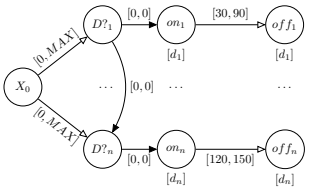
We first present two representations for the general structure of the network, which is heavily influenced by the way device events are modelled. The first model uses conditionality to capture the possibility of an event happening, and the second does not, instead attempting to be as simple as possible.

4.2.1) Conditional model: The relation between on_i and off_i is a well-defined contingent link: there is some estimate of the duration of the activation cycle, but the assignment of off_i is not under the Agent's control. However, on_i is not controllable either. Furthermore, nothing accurate can be said about its constraint with regards to another time-point, or which of the devices is expected first.

We can capture the lack of control over the time-point with yet another contingent link, this time one with the interval $[0, MAX]$. We can use an observation node $D?$ to model the uncertainty of which device activates next. Figure 3a shows an example of this.



(a) One observation for n device events cannot capture all options.

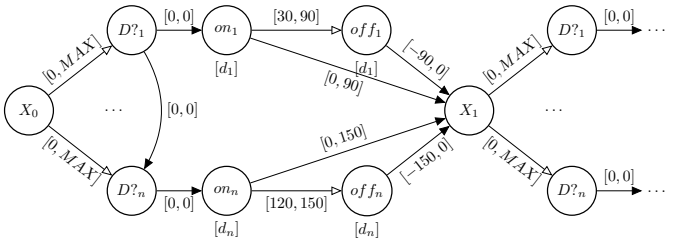


(b) Using multiple observation nodes.

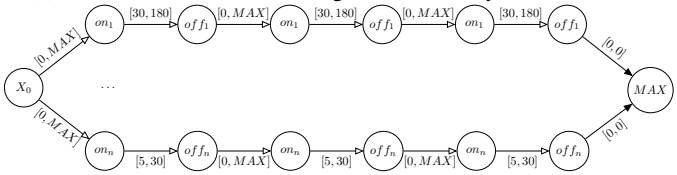
Figure 3: Modelling the device activation events

The observed proposition is a boolean value, meaning that it cannot express $|D|$ different observations. To solve this, we can instead define a separate proposition for each device. We observe these at the same instant, and set all propositions to false, except for the device that activated. This network is shown in Figure 3b.

Lastly, we must repeat this for the next device event, as shown in Figure 4a. The next event can occur in two ways. It can occur (1) after off_i or (2) before off_i (but still after on_i). In the first case, the solution is trivial: we simply connect off_i to the next dummy node X_1 and repeat the graph structure. In the second case, we get overlap between the structures,



(a) Conditional model including simultaneously active devices.



(b) Non-conditional model using sliding window.

Figure 4: Two ways of modelling device activations.

because the device from the first loop is still active, and its deactivation time-point has not been executed yet. Using d_1 as an example, we solve this by introducing a negative constraint between from off_i to X_1. In the worst case, a new device activates immediately after the first. X_1 is then executed to enter the second repetition, but we still have to wait at most 90 minutes for d_1 to turn off. In the best case, $off_i = X_1$, meaning that no device turned on during the activation cycle of d_1.

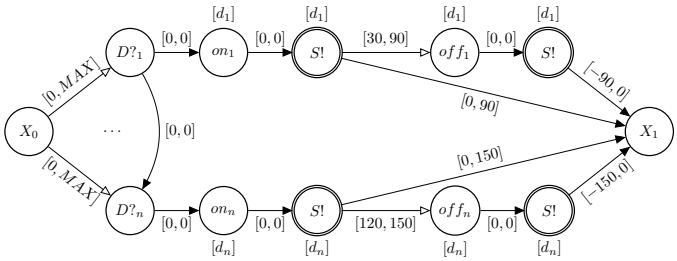
4.2.2) Non-conditional model: A second approach, which will prove to have some merit, is to not consider any conditionality. We instead assume that every device event occurs at some point, however far away that may be in the future. If it happens outside the sliding window, that is not a problem, because the window will progress and eventually reach the event.

We do not create a repetition of activation cycles, but instead describe all events that are expected within the horizon. Each path from X_0 to MAX describes the activation cycles of one device and is independent of the other paths.

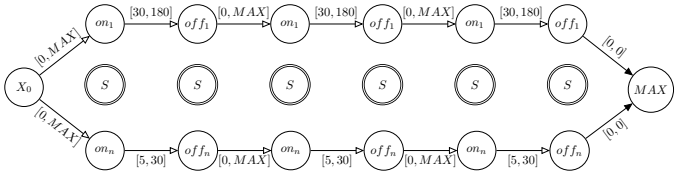
4.3. Modelling channel switch decisions

To enable reasoning about switching, we introduce extra time-points that denote the switch. We first make some observations about how the device events affect the choice for active channel. Each device activation increases the self-consumption on its respective channel. This means that the only resulting action (other than not doing anything) of a device activation might be to *switch to that channel*. Inversely, when a device turns off, the self-consumption of its respective channel decreases, and a resulting action can be to switch to another channel. Note that because not doing anything is always an option, the deactivation of a device can effectively cause a switch to any channel.

Following a greedy logic, we see that a switch should always occur immediately after observing a device event. We therefore add switch nodes exactly after the device events.



(a) Switch decision nodes ($S!$) in one event cycle of the conditional model.



(b) Unconstrained switch nodes in the non-conditional model.

Figure 5: Adding sets of $|C|$ switch nodes to both models.

For the conditional model we use *decision nodes*, as introduced in Definition 5. We quickly find that we need a separate decision node for each channel, just as with the observation nodes for device events. For sake of readability, we denote the set of $|C|$ decision nodes at each location with just one node $S!$. The result is shown in Figure 5a.

For the non-conditional model, we do not have the concept of a decision. Instead, we must pre-define the decision in each node, so that executing a node is equivalent to switching to its associated channel. We get, similar to the set of decision nodes in Figure 5a, a set of switch nodes that each represent a switch to one channel. However, contrary to the conditional model, we cannot put these sets after every device event, because in a non-conditional STP, all nodes are executed. What we then get is an unconstrained (unconnected) set of nodes, as shown in Figure 5b. While this is valid, we lose some control, since we no longer have $[0, 0]$ constraints between a device event and a switch. We will need to add other logic that reasons about their execution time.

4.4. Decision-making

While the moment of switching can be greedily decided, the less trivial question is what channel to switch to. This choice depends on a lot of factors, but it is not temporally constrained: the system can switch to any channel at any time. Three major factors that influence the optimal decision are (1) the PV production forecast, (2) the energy consumption in each channel, and (3) the balance of the energy distribution between the channels.

We now discuss two possible ways of deciding an optimal choice and apply these to both models.

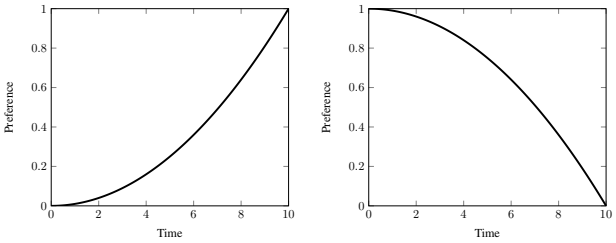
4.4.1) Temporal constraint relations: The three factors described above are not temporal constraints, but they can be described as a function over time. Based on the PV production forecast, we can estimate how much time a channel needs to be active to catch up with the energy distribution. Based on the active devices, we can estimate for each household how long this consumption will stay active.

We can use this temporal information to create temporal constraints in the network. This is most clear in the non-conditional model, since we can constrain the switch to one channel by the time difference with some other channel.

In the conditional model, we have already fixed the time values for the switches, making this approach less feasible. However, we can still apply this logic across the event cycles. What we get are constraints that cross the event cycle boundary. In the next cycle, the decision for a channel can then only be made if the constraints with previous decisions are satisfiable. To prevent the network from always becoming inconsistent, this approach requires the notion of *conditional constraints*, which was introduced in the framework of the *streamlined CSTN* [18] and is a rather simple variation of the original CSTN model. In this framework the edges are labelled instead of the nodes. If so desired, the original node-labelled definition can be used, but this requires some dummy nodes to express such a conditional constraint.

4.4.2) Preference function: As described in Definition 8, a soft constraint contains a preference function that can describe better and worse (temporal) assignments. We would like a similar concept, but instead for assigning a truth value to a decision proposition.

We can use the sliding window concept to choose to 'delay' a switch node. The channels that are not preferred get a low preference value in the near future, but a higher preference in the later future. The most preferred channel gets a high preference value at $t = 0$. The general idea of such functions is shown in Figure 6.



(a) Preference to switch later. (b) Preference to switch now.

Figure 6: Temporal preference functions that determine a switch decision.

This approach only works when using the non-conditional model, because the execution moment of those switch nodes is not fixed yet. In the conditional model, we execute all switch decision nodes, so reasoning about their temporal execution does not work. We can, however, still use temporal preference to differentiate our choices. Rather than applying a preference function on the decision nodes, we introduce an extra node after each switch decision. This 'profit' node is only executed if the corresponding channel was activated, that is: the corresponding proposition is true (and the propositions

for other switches are not). We can then apply the preference function for that channel to this profit node. We demonstrate such a system for one set of three decision nodes.

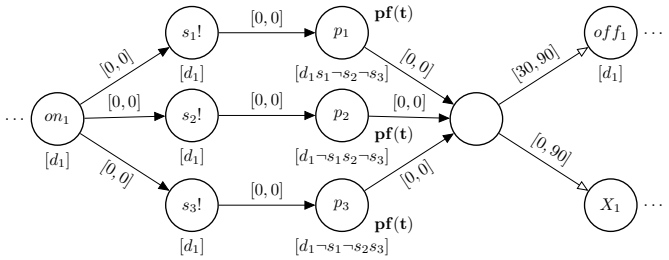


Figure 7: Profit nodes (p_i) for one set of switch decisions after one event (on_1). Each has a preference function $\mathbf{pf(t)}$.

5. Comparison & Limitations

We now compare the various options, which are summarized in Table II, along with the class that implements the necessary concepts to define them. As can be seen, no class actually implements conditions and preferences, making this option impossible.

Device event model	Decision-making	Class	Complexity
Non-conditional	Temporal constraints	STPU	P
Non-conditional	Preference function	STPPU	P
Conditional	Temporal constraints	CSTNUD[1]	EXP [14]
Conditional	Preference function	-	-

Table II: Proposed solutions and problem class they belong to.

5.1. Comparison of network models

Disregarding the decision-making, the conditional model can be implemented using a CSTNUD, while the non-conditional version can be modelled by STPU. The non-conditional model is therefore less complex, as practical algorithms for STPU do exist, in contrast to CSTNUD. The downside of the STPU network for our problem is its lack of conditional expressions, which are useful throughout the network to reason about history in a more intuitive manner. The lack of conditionality also leads to an awkward way of reasoning about the future, since we effectively keep widening the constraints of future device events by using a sliding window. Another example of these limitations is the inability to create a $[0, 0]$ constraint between switches and device events in an STPU.

CSTNUD is by definition the more expressive class, and provides concepts that are key in our problem. This makes the problem more intuitive to model, and allows us to express more accurate and detailed constraints. However, even this model becomes unintuitive when trying to express conditions that are non-boolean, such as in our motivating problem. Furthermore there are a few practical issues. Firstly, CSTNUD with preferences does not exist and the extension is not trivial. Furthermore CSTNUD has no sound and complete propagation-based algorithm. It should be noted that sound but incomplete propagation rules do exist, and that the dynamic controllability of a CSTNU(D) can be translated into a reachability question by means of a timed-game automaton (TGA), for which sound and complete solutions exist, but are too difficult to solve in practical settings.

[1]The streamlined version by [18] is preferred.

5.2. Comparison of decision-making strategies

The approach of using temporal constraints is beneficial, because it uses the strength of TCSPs, namely to reason about temporal constraints. However, the abstraction leads to some problems. The most problematic is that using these constraints can easily lead to an inconsistent network. We can only extract an interval based on a target amount of PV production or self-consumption. If this amount is unreachable, no switch whatsoever will be allowed. This type of problem is normally solved by lowering the target optimality, and thus loosening the constraints, until a consistent solution is found. However, since the constraints are independent of each other, and the global optimality is based on their combination, there is no clear strategy for relaxing the constraints.

A second problem is that defining the constraints itself is non-trivial as well. To a degree, deciding on the optimal set of constraints is already solving the scheduling problem.

Using a preference function is more intuitive, since it makes a distinction between hard constraints and soft constraints, and the algorithms solving networks with soft constraints do have the ability to iteratively lower the target preference value. Furthermore, a function is better able to describe the non-temporal aspects of the optimization, since it does not to translate them into constraint intervals.

The downside to this approach is that defining the function becomes a complicated task. Almost all of the optimality is contained in a function that must be designed by a knowledge engineer, while the network does no reasoning about the optimality of a decision, except picking the highest-valued solution. Also, we see that the soft constraints do not directly express the problem of interest, namely the value of decisions, making them still unintuitive to use.

5.3. General problems in modelling

We can state a few general problems that arise when using TCSPs, based on the above observations.

Firstly, we see that TCSPs are only able to express well-defined uncertainties. As the uncertainty becomes vaguer, such as when little can be said about what exactly is going to happen, or about the temporal relations thereof with other events, it becomes more difficult to model it.

Secondly, we find that expressing optimality in TCSP is difficult. The main purpose of a TCN is to check its consistency, which only decides whether any solution exists (and gives it). The model only provides little notion of optimality at all, and only does so for the assignment of time-points.

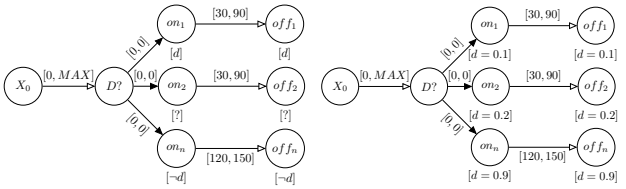
For this reason, we can conclude that TCSP does not provide the required notions for expressing the motivating problem.

6. Suggested Extensions

We now present two possible extensions to the TCSP model: *non-boolean conditions* and *conditional preference*. These aim to solve the problems observed in the previous section. We explain their concepts and give a definition, but do not discuss how these can be added to existing models or algorithms, as that is outside the scope of this paper.

6.1. Non-boolean conditions

We first address the issue observed in section 4.2.1 that an observation can only distinguish two situations. We solved this by grouping multiple observations together. The same problem occurs with decision nodes, for the same reason. In CTPP, defined by [11], the boolean propositions are replaced by fuzzy propositions, which can take a truth value anywhere in the domain $[0, 1] \in \mathbb{R}$. We use this concepts as inspiration, and also replace boolean propositions with fuzzy propositions. However, rather than expressing a finer-grained truth degree of some proposition, we discretize the range to express multiple, but exclusive disjunctive propositions. This applies to the problem of observing the activation of $|D|$ devices, since only one device can turn on at one time (and the future devices that turn on are observed by the next cycle). The extension is shown in Figure 8 for the original problem. We define the same idea for non-boolean decisions.



(a) A single boolean observation cannot capture all possibilities.

(b) A single non-boolean observation captures any finite number of possibilities.

Figure 8: Original boolean observations and discretized fuzzy observations.

The benefit of this is mostly semantics and elegance. A fuzzy condition can express exclusive disjuncts, and prevents the need for $|D|$ nodes. It does not provide additional expressiveness, since multiple boolean conditions can express a single fuzzy condition too. However, we should observe that the lack of multiple nodes can have an impact on preference functions and temporal constraints. This impact can be positive, requiring only a single function over multiple ones, or worse, not being able to bind a temporal constraint to a specific observation.

6.2. Conditional Preference

the second issue is a lack of reasoning about decisions. Since the original temporal preference functions only define preference over time, they do not have direct influence on the decision that is made. It can however be that one decision is preferred over the other for non-temporal reasons. To express such cases, we define the notion of conditional preference, that is similar to the way the temporal preference function is defined.

Definition 10. A conditional preference function is a function $cp : [0, 1] \rightarrow [0, 1]$ associated with a decision node $D!$. The preference function takes any valid decision option of $D!$ and returns a preference value for this decision.

In this problem, we could exploit the greedy temporal relation of switching decisions with device events to achieve the result of conditional preference. However, when the decision nodes are not so tightly constrained—a situation that is common in TCSPs—we cannot do this. This extension then enables expressions that are otherwise not possible.

7. Related Work

Using TCSP to optimize smart grid scheduling has, to the best of our knowledge, not been attempted before. However, STNU optimization has been done with various notions of optimality [19] and this has led to reformulating the optimization problem as MILP, MINLP, or Conflict-Directed Search. There is a broad range of methods for decision making under uncertainty in smart grids [20]. Globally speaking, some of the directions of research are probabilistic analysis, possibilistic analysis, and robust optimization.

Different definitions of conditionality exist, that we have not discussed here, such as the Simple Temporal Network with Alternatives (STNA) [21]. It uses a concept of branching to express different choices, which is unsuitable for the less well-defined decision paths in our problem, and does not deal with external factors (observations). The propositional system of CTP provides much more expressiveness.

A different branch of TCSP research explores the flexibility and relaxation strategies for decisions in the network [22] [23]. These models mostly deal with decision-making in controllable environments, which is not relevant to our problem, because of its high uncontrollability.

Various other approaches to temporal reasoning and scheduling exist, that do not use temporal constraint networks, but instead use logical languages, such as PPLAN [24], SMTL [25], and PDDL [26]. We do not consider these in this paper, but future work could investigate their merit for expressing detailed information and reasoning about it.

8. Conclusions

We have attempted to model a smart grid problem in a TCSP in various ways. This has illuminated some limitations in the existing models with regards to modelling uncertainty and defining optimality. We have discussed advantages and drawbacks of the different approaches, concluding that a non-conditional model is not able to give an accurate enough description of the problem, but a conditional model is too complex to be practically solved. Furthermore, using temporal constraints to enforce preferences in the network is undesirable, but using preference functions still requires substantial effort and does not fully capture the exact notion of preference

that we look for. This leads us to conclude that TCSP is not suitable for our problem, and in general was not meant for the kind of problems that often occur in smart grids.

To solve these issues, we have defined two extensions: non-boolean (fuzzy) conditions that can express more than one proposition when the propositions are exclusive disjunctive, and conditional preferences that define how optimal one decision is over another.

While these have merit in this problem and also in other problems, we are unsure whether they are sufficient to enable modelling the motivating problem in a TCSP. This is because of various properties of the TCSP model.

Firstly, temporal networks require a well-defined set of events. In the case that there are uncertainties, their possible outcome and the resulting effect on the network must be clearly defined. When the future is too vague, the network has no ways to reason about it.

Secondly, TCSP is primarily intended for constraint satisfaction. In the context of scheduling this means guaranteeing correctness of the schedule. However, in our problem the correctness is not an issue, but optimality is, something that is only a limited secondary concept in temporal constraint problems.

Our motivating problem furthermore does not deal with complicated temporal assignments, something that TCSN consistency solving automatically produces. Instead its most complicated aspect is decision-making, something that is much less sophisticated in TCSP when not clearly defined by temporal relations.

While we cannot evaluate the merit of detailed information in smart grid scheduling, we can see that TCSP does not provide a straightforward answer for solving problems in this domain.

9. Future Work

For future work we suggest to implement the proposed extensions, and look at optimization models that can be used in the preference function. Based on those results, a more definitive conclusion can be made regarding the feasibility of using temporal networks for smart grids problems.

Other work can be done in finding different models that can use detailed information, but are also able to handle high levels of uncertainty and uncontrollability.

References

[1] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.

[2] T. Vidal and H. Fargier, "Handling Contingency in Temporal Constraint Networks: from Consistency to Controllabilities," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 11, no. 1, pp. 23–45, 1999.

[3] I. Tsamardinos and M. E. Pollack, "Efficient solution techniques for disjunctive temporal reasoning problems," *Artificial Intelligence*, vol. 151, no. 1-2, pp. 43–89, 2003.

[4] L. Khatib, P. Morris, R. Morris, and F. Rossi, "Temporal constraint reasoning with preferences," in *IJCAI International Joint Conference on Artificial Intelligence*, 2001, pp. 322–327.

[5] P. Morris, N. Muscettola, and T. Vidal, "Dynamic Control Of Plans With Temporal Uncertainty," no. November 2016, 2001.

[6] P. H. Morris and N. Muscettola, "Temporal Dynamic Controllability Revisited," *Aaai*, vol. 94043, pp. 1193–1198, 2005.

[7] F. Rossi, K. B. Venable, and N. Yorke-Smith, "Uncertainty in Soft Temporal Constraint Problems: A General Framework and Controllability Algorithms for The Fuzzy Case," *Journal of Artificial Intelligence Research*, vol. 27, pp. 617–674, 2006.

[8] I. Tsamardinos, T. Vidal, and M. E. Pollack, "CTP: A New Constraint-Based Formalism for Conditional, Temporal Planning," *Constraints*, vol. 8, no. 4, pp. 365–388, 2003.

[9] C. Comin and R. Rizzi, "A Singly-Exponential Time Algorithm for Dynamic Consistency of Conditional Simple Temporal Networks," 2003.

[10] L. Hunsberger, R. Posenato, and C. Combi, "A Sound-and-Complete Propagation-Based Algorithm for Checking the Dynamic Consistency of Conditional Simple Temporal Networks," in *TIME 2015: 22nd International Symposium on Temporal Representation and Reasoning (TIME 2015)*, 2015, pp. 4–18.

[11] M. Falda, F. Rossi, and K. B. Brent Venable, "Fuzzy conditional temporal problems: Strong and weak consistency," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 5, pp. 710–722, 2008.

[12] B. Peintner and M. E. Pollack, "Low-cost Addition of Preferences to DTPs and TCSPs," *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pp. 723–728, 2004.

[13] L. Hunsberger, R. Posenato, and C. Combi, "The Dynamic Controllability of Conditional STNs with Uncertainty," *Proceedings of the Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) at ICAPS-2012*, pp. 1–8, 2012.

[14] M. Jurdziński and A. Trivedi, "Reachability-time games on timed automata," *Automata Languages and Programming*, pp. 838–849, 2009.

[15] M. Cairo, C. Combi, C. Comin, L. Hunsberger, R. Posenato, R. Rizzi, and M. Zavatteri, "Incorporating Decision Nodes into Conditional Simple Temporal Networks," *Leibniz International Proceedings in Informatics, LIPIcs*, vol. 90, no. 9, pp. 1–9, 2017.

[16] M. Zavatteri, "Conditional Simple Temporal Networks with Uncertainty and Decisions," *Drops-Idn/7916*, vol. 90, no. 23, pp. 1–17, 2017.

[17] A. Cimatti, L. Hunsberger, A. Micheli, R. Posenato, and M. Roveri, "Dynamic controllability via Timed Game Automata," *Acta Informatica*, vol. 53, no. 6-8, pp. 681–722, 2016.

[18] M. Cairo, L. Hunsberger, R. Posenato, and R. Rizzi, "A Streamlined Model of Conditional Simple Temporal Networks – Semantics and Equivalence Results," *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, vol. 90, no. 10, pp. 1–10, 2017.

[19] J. Cui, P. Yu, C. Fang, P. Haslum, and B. C. Williams, "Optimising Bounds in Simple Temporal Networks with Uncertainty under Dynamic Controllability Constraints." *ICAPS 2009: Proceeding of 19th International Conference on Automated Planning and Scheduling*, pp. 52–60, 2015.

[20] A. Soroudi and T. Amraee, "Decision making under uncertainty in energy systems: State of the art," pp. 376–384, 2013.

[21] R. Barták, O. Čepek, and P. Surynek, "Modelling alternatives in temporal networks," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling, CI-Sched 2007*, 2007, pp. 129–136.

[22] E. Timmons and B. C. Williams, "Enumerating Preferred Solutions to Conditional Simple Temporal Networks Quickly Using Bounding Conflicts," in *AAAI Workshop on Planning, Search, and Optimization*, 2015, pp. 100–101.

[23] P. Yu and B. Williams, "Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution," in *IJCAI International Joint Conference on Artificial Intelligence*, 2013, pp. 2429–2436.

[24] M. Bienvenu, C. Fritz, and S. McIlraith, "Planning with Qualitative Temporal Preferences," *Proceedings of the Tenth International Conference on the Principles of Knowledge Representation and Reasoning*, no. Reiter, pp. 134–144, 2006.

[25] R. Luo, R. Valenzano, Y. Li, J. C. Beck, and S. A. Mcilraith, "Using Metric Temporal Logic to Specify Scheduling Problems," no. Kr, pp. 581–584, 2016.

[26] J. Benton, A. Coles, and A. Coles, "Temporal Planning with Preferences and Time-Dependent Continuous Costs," *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 2–10, 2012.