Time Resource Networks

Paper 487

Abstract

The problem of scheduling under resource constraints is widely applicable. One prominent example is power management, in which we have a limited continuous supply of power but must schedule a number of power-consuming tasks. Such problems feature tightly coupled continuous resource constraints and continuous temporal constraints.

We address such problems by introducing the Time Resource Network (TRN), an encoding for resource-constrained scheduling problems. The definition allows temporal specifications using a general family of representations derived from the Simple Temporal network, including the Simple Temporal Network with Uncertainty, and the probabilistic Simple Temporal Network (Fang et al. (2014)).

We propose two algorithms for determining the consistency of a TRN: one based on Mixed Integer Programing and the other one based on Constraint Programming, which we evaluate on scheduling problems with Simple Temporal Constraints and Probabilistic Temporal Constraints.

1 Introduction

Temporal Networks scheduling algorithms support diverse formulations useful in modeling practical problems. Examples include dynamical execution strategies based on partial knowledge of uncertain durations, or strategies upper-bound the probability of failing to satisfy probabilistic constraints. However, it is not obvious how to apply them in scenarios with resource usage constraints. Those kind of scenarios, on the other hand are investigated in Operations Research literature and known as project scheduling. Time Resource Network attempt to narrow the gap between those two bodies of work, which so far remained mostly independent.

Let's consider the following Smart House scenario. We have $150W$ generator which is available. We know that the user comes back from work at some time defined by a gaussian distribution $N(5pm, 5m)$. Moreover we know that sun sets at time defined by $N(7pm, 1m)$. We would like to meet the following constraints with the overall probability at least 98%:

- Wash clothes (duration: $2h$, power usage: $130W$) before user comes back from work
- Cook dinner (duration: $30m$, power usage: $100W$) ready within 15 minutes of user coming back from work
- Have the lights on (power usage: $80W$) from before sunset to at least midnight.
- Cook a late night snack (duration: $30m$, power usage: $20W$) between 10pm and 11pm.

While probabilistic constraints can be straightforwardly modeled using probabilistic Simple Temporal Networks [Fang *et al.*, 2014], there is no known algorithm that can jointly model the resource constraint. In this publication we introduce Time Resource Networks, which are capable of finding schedules for this type of scenarios. Solution to the example described above is presented on fig. 1. In meets the constraints with $99,7\%$ probability.

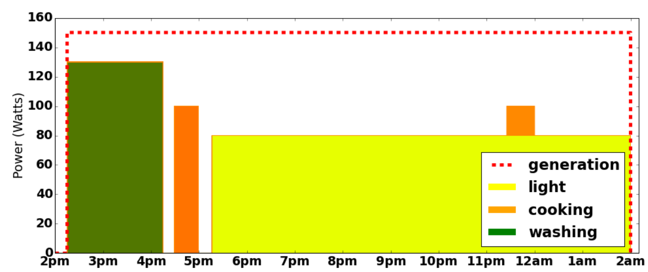

Figure 1: Depiction of solution to TRN spanning a pSTN.

2 Related Work

One of the earliest mentions of a scheduling problem being solved in an algorithmic fashion can be found in [Johnson, 1954], although there's evidence that the problem was already considered in unpublished versions of [Bellman, 1956]. This publication considers the following statement of scheduling problem. We have n items and m stages and $A_{i,j}$ denoting the time for i-th item to be processed by stage j. All the items must be processed by different stages in order (for example first stage is printing of a book and second stage is binding). The publication considers $m = 2$ and $m = 3$ and arrives at the solution that *"permits one to optimally arrange twenty production items in about five minutes by visual inspection"*. It turns out that the solution to the problem for $m \geq 3$ is NP-hard ([Garey *et al.*, 1976]). In [Wagner, 1959] an Integer Pro-

gramming solution to the scheduling problem was presented, with a comment stating that it *"is a single model which encompasses a wide variety of machine-scheduling situations"*. In [Pritsker *et al.*, 1969], a generalization of scheduling problems is considered, which allows for multiple resource constraints. However, the proposed solution uses a discrete time formulation, which, depending on required accuracy, can substantially decrease performance. In 1988 a technique was proposed which can handle resource constraints and continuous time ([Bartusch *et al.*, 1988]). The proposed approach can be thought of as resource constrained scheduling over Simple Temporal Networks (STN).
In [Dechter *et al.*, 1991], a notion of Simple Temporal Problem was introduced which allows one to solve problems with simple temporal constraints of form $l \leq t_y - t_x \leq u$. This concept was later extended with various more sophisticated notions of temporal constraints. [Vidal and Ghallab, 1996] defined an uncertain temporal constraint, where the duration between two time events can take a value from an interval $[l, u]$, which is unknown during the time of scheduling (uncertain duration constraints). [Morris *et al.*, 2001] describes a pseudopolynomial algorithm for handling uncertain duration constraint, where we are allowed to make a scheduling decisions based on knowledge of uncertain durations from the past (Dynamic controllability). The algorithm is later improved to polynomial complexity ([Morris and Muscettola, 2005]). Finally, [Fang *et al.*, 2014] provides a non-linear optimization based solver for uncertain temporal constraints where the duration of the constraint can come from arbitrary probabilistic distribution.

3 Problem statement

In this section we introduce a novel formulation - Time Resource Network (TRN). All the results presented in this paper can be extended to multiple different type of resources being constrained at the same time (electricity, water, fuel, cpu time, memory etc.), but to simplify the notation we will assume that only one type of resource is constrained. Additionally, we only consider the problem of consistency, but the techniques presented in this paper can be extended to handle objective optimization over constrained schedules.

3.1 Abstract Temporal Network

TRN's definition supports many different temporal networks. To capture only the relevant properties, we define the notion of Abstract Temporal Network as $ATN = (events, extend)$:

1. `events(ATN)`, returns a set of events in ATN
2. `extend(ATN,` $\{stc_1, ...stc_n\}$`)`, which takes ATN and a set of simple temporal constraints ([Dechter *et al.*, 1991]) spanning `events(ATN)`, and returns another ATN', such that there exists a schedule satisfying $TC(ATN')$ if and only if there exists a schedule satisfying $TC(ATN)$ and obeying set of simple temporal constraint $\{stc_1, ...stc_n\}$. TC is a notion of temporal consistency described in section 3.3.

As the following section describes in detail we will use `extend` to encode resource constraints over `events`.

3.2 Schedule

A schedule s : `events(ATN)` $\rightarrow \mathbb{R}$ is a mapping from events in ATN to their execution times.

3.3 Temporal Consistency

For an ATN we define a predicate $TC_s(ATN)$, which means that ATN is **temporally consistent** under schedule s. TC_s is true if schedule s satisfies all the constraints of the ATN (what that means precisely depends on the ATN - we only require for it to be verifiable). We say that ATN is temporally consistent (denoted by $TC(ATN)$), when there exists at schedule s such that $TC_s(ATN)$.

3.4 Time Resource Network

A Time Resource Network is described by a tuple $TRN = (ATN, R)$, where ATN is an Abstract Temporal Network and $R = src_1, ..., src_n$ is a set of **simple resource constraints**, each of which is a triplet (x, y, r), where $x, y \in$ `events(ATN)` and $r \in \mathbb{R}$ is the amount of resource, which can be positive (consumption) and negative (generation). Given a schedule s for any time $t \in \mathbb{R}$ we define **resource usage** for $src = (x, y, r)$ as:

$$u_s(src, t) = \begin{cases} r & \text{if } s(x) \leq t < s(y) \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, simple resource constraint encodes the fact that between time $s(x)$ and $s(y)$ resource is consumed (generated) at the rate $|r|$ per unit time for positive (negative) r.
Our notation is inspired by [Bartusch *et al.*, 1988]. The authors have demonstrated that it is possible encode arbitrary piecewise-constant resource profile, by representing each constant interval by a simple resource constraint and joining ends of those intervals by simple temporal constraints.

3.5 Resource consistency

For a schedule s we define a **net-usage** of a resource at time $t \in \mathbb{R}$ as:

$$U_s(t) = \sum_{\forall_{src_i \in R}} u_s(src_i, t)$$

R is the set of all the resource constraints. We say that the network is **resource consistent** under schedule s when it satisfies predicate $RC_s(TRN)$, i.e.

$$\forall_{t \in \mathbb{R} - C} . U_s(t) \leq 0 \tag{1}$$

where C is some *finite* set of real numbers. Intuitively, it means that resource is never consumed at a rate that is greater than the generation rate. Set C is introduced to make it easier to prove certain properties, but is of no practical significance - notice that regardless of the contents of C above statement is true 100 % of time - there exists no positive length interval where $U_s > 0$. We say that TRN is resource consistent, if there exists s, such that $RC_s(TRN)$ is true.

3.6 Time-resource consistency

$TRN = (ATN, R)$ is **time-resource consistent** if there exists a schedule s such that $RC_s(TRN) \wedge TC_s(ATN)$. Determining whether a TRN is time-resource consistent is the central problem addressed in this publication.

3.7 Properties of TRN

Before we proceed to describe algorithms for determining time-resource consistency it will be helpful to understand some properties that apply to every TRN.

Lemma 3.1. *For a TRN a schedule s is resource consistent if and only if*

$$\forall_{e \in \texttt{events(ATN)}} \lim_{\epsilon \to 0} U_s(s(e) + \epsilon) \leq 0 \qquad (2)$$

i.e. resource usage is not non-positive a moment after all of the scheduled events.

Proof. $\Rightarrow$ Follows from definition of resource-consistency.
$\Leftarrow$ We say a time point $t \in \mathbb{R}$ is scheduled if there exists an event $x \in$ `events(ATN)` such that $t = s(x)$. Assume for contradiction, that the right side of the implication is satisfied, but the schedule is not resource consistent. That means that there exists a time point t_{danger} for which $U_s(t_{danger}) > 0$. We will only consider the case where t_{danger} is **not** scheduled (because there are finitely many scheduled time points, we can consider them members of C). Let t_{before} be the highest scheduled (so $t_{before} = s(e_{before})$ for some $e_{before} \in$ `events(ATN)`) time point that is smaller than t_{danger}. Notice that if no such time point existed, that would mean that there is no resource constraint (x, y, r) such that $s(x) \leq t_{danger} < s(y)$, so $U_s(t_{danger}) = 0$. We can therefore assume that t_{before} exists. Notice that by definition of t_{before} and simple resource constraints, $U_s(t)$ for $t_{before} < t \leq t_{danger}$ is constant, therefore $U_s(t_{danger}) = \lim_{\epsilon \to 0} U_s(s(e_{before}) + \epsilon) > 0$. Contradiction. □

Corollary 3.1.1. *Given a TRN and two schedules A and B where all events occur in the same order, A is resource-consistent if and only if B is resource-consistent.*

Proof. Notice that if we move execution time of arbitrary event, while preserving the relative ordering of time points, then net resource usage moment after that event will not change (as the $U_s(t)$ between the neighboring events remains constant). Therefore by lemma 3.1 we can transform schedule A into schedule B while preserving resource consistency. □

4 Approach

In this section we present two approaches for determining time-resource consistency of TRN. One of them is using Mixed Integer Programming (MIP) and the other is using Constraint Problem (CP) formulations. For both algorithm the following definitions will be useful. Let's take a $TRN = (ATN, R)$ where $R = src_1, ..., src_n$ and $src_i = (x_i, y_i, r_i)$ as defined in section 3.4. Let's denote all the events relevant for resource constraints as $RE \subseteq events(ATN)$, i.e.

$$RE = \{x_i | (x_i, y_i, r_i) \in R\} \cup \{y_i | (x_i, y_i, r_i) \in R\}$$

Additionally, let's introduce resource-change at event $e \in events(ATN)$ as:

$$\Delta(e) = \sum_{(x_i, y_i, r_i) \in R, x_i = e} r_i + \sum_{(x_i, y_i, r_i) \in R, y_i = e} -r_i$$

Intuitively $\Delta(n)$ is the amount by which resource usage changes after time $s(n)$ under schedule s.

4.1 Mixed Integer Programming based algorithm

Mixed Integer Programming ([Markowitz and Manne, 1957]) is a very natural way of expressing scheduling problems. It's flexibility and efficiency causes many researchers to choose this method to tackle scheduling problems. In this section we present a way to formulate TRN as a MIP problem. The technique is very similar to the ones used in state of the art solvers for general scheduling [Patterson, 1984] [Bartusch *et al.*, 1988]. Therefore the purpose of this section is not to introduce a novel approach, but to demonstrate that those algorithms are straightforward to express using TRN formulation. Let $TC - fromulation(ATN)$ be a MIP-formulation that is consistent if an only if $TC(ATN)$. For some types of ATN such a formulation might not exist and in those cases MIP-based algorithm cannot be applied.
We will use the following formulation:

$$\forall_{e \in events(ATN)}. \qquad 0 \leq e \leq M \qquad (3)$$

$$\forall_{e_1, e_2 \in RE, e_1 \neq e_2}. \qquad e_1 - e_2 \geq -x_{e_1, e_2} M \qquad (4)$$

$$\forall_{e_1, e_2 \in RE, e_1 \neq e_2}. \qquad e_1 - e_2 \leq (1.0 - x_{e_1, e_2}) M \qquad (5)$$

$$\forall_{e_1, e_2 \in RE, e_1 \neq e_2}. \qquad x_{e_1, e_2} + x_{e_2, e_1} = 1 \qquad (6)$$

$$\forall_{e_1, e_2 \in RE, e_1 \neq e_2}. \qquad x_{e_1, e_2} \in \{0, 1\} \qquad (7)$$

$$\forall_{e_1 \in RE}. \qquad \sum_{e_2 \in RE} x_{e_2, e_1} \Delta(e_2) \leq 0 \qquad (8)$$

$$\text{TC-fromulation(ATN)} \qquad (9)$$

Variable M denotes the time horizon, such that all the variables are scheduled between 0 and M. This definition is imposed in eq. 3. Variables x_{e_1, e_2} are order variables, i.e.

$$x_{e_1, e_2} = \begin{cases} 1 & \text{if } s(e_1) \leq s(e_2) \\ 0 & \text{otherwise} \end{cases}$$

Equations 4, 5, 6, 7 enforce that definition. In particular equations 4, 5 enforce the ordering using big-M formulation that is correct because of time horizon constraint. In theory eq. 6 could be eliminated by careful use of ϵ (making sure no two timepoints are scheduled at exactly the same time), but we found that in practice they result in useful cutting planes that decrease the total optimization time. Equation 8 ensures resource consistency by lemma 3.1. Finally eq. 9 ensures time consistency.
Solving that Mixed-Integer Program will yield a valid schedule if one exists, which can be recovered by inspecting values of variables $t \in events(ATN)$.

4.2 Constraint Programming based algorithm

The downside of MIP approach is the fact that the ATN must have a MIP formulation (e.g. pSTN does not have one). In this section we present a novel CP approach which addresses those concerns. The high level idea of the algorithm is quite simple and is presented in algorithm 1. In the second line, we iterate over all the permutations of the events. On line 3 we use `resource_consistent` function to check resource consistency, which by corollary 3.1.1 is only dependent on the chosen permutation. On line four we use TC checker to determine if network is time consistent - the implementation depends on ATN and we assume it is available. Function *encode_as_stcs* encodes permutation using simple temporal constraints. For example if $\sigma(1) = 2$ and $\sigma(2) = 1$

and $\sigma(3) = 3$, then we can encode it by two STCs: $2 \leftarrow 1$ and $1 \leftarrow 3$.

Data: TRN
Result: true if TRN=(ATN, R) is time-resource-consistent

$N \leftarrow$ `events(ATN);`
for $\sigma \leftarrow$ *permutation of* N **do**
  if `resource_consistent(R, σ)` **then**
    if *TC(*`extend(ATN, encode_as_stcs(σ)))` **then**
      return true;
    end
  end
end
return false;

Algorithm 1: Time-resource-consistency of a TRN

The implementation of `resource_consistent` follows from lemma 3.1 and is straightforward - we can evaluate $\lim_{\epsilon\to 0} U_s(s(e) + \epsilon)$ for all events $e \in RE$ only knowing their relative ordering, if it is always non-positive then we return true.

To improve the performance w.r.t algorithm 1 we use off-the-shelf constraint propagation software (PyConstraint). Let's consider $RE = e_1, ..., e_N$. We define a problem using N variables: $x_1, x_2, ..., x_N \in \{1, ..., N\}$, such that $x_j = i$ if e_i is j-th in the temporal order, i.e. $x_1, ..., x_N$ represent the permutation σ. We used the following pruners which, when combined, make the CP solver behave similarly to algorithm 1, but ignoring some pruned permutations:

- **all_different_constraint** - ensure that all variables are different, i.e. they actually represent a permutation. This is standard constraint available in most CP software packages.
- **time_consistent** - making sure that the temporal constraints implied by the permutation are not making the ATN inconsistent. Even when the variables are partially instantiated, we can compute a set of temporal constraints implied by the partially instantiated permutation. For example if we only know that $x_1 = 3$, $x_5 = 2$ and $x_6 = 5$, it implies $e_5 \leq e_1 \leq e_6$.
- **resource_consistent** - ensure that for all $e_1, ..., e_n \in RE$, resource usage just after e_i is non-positive. Even if the order is partially specified we can still evaluate it. A subtlety which needs to be considered is that we need to assume that all the events for which x_i is undefined and which are generating ($\delta(e_i) < 0$) could be scheduled before all the points for which order is defined. For example if $n = 4$ and $\Delta(e_1) = 4$, $\Delta(e_2) = -6$, $\Delta(e_3) = 3$, $\Delta(e_4) = 4$ and we only know that $x_1 = 3$, $x_3 = 2$, then we have to assume that all the generation happened before the points that we know, i.e. initially resource usage is -6, then after e_3 is is -3, and after e_1 it is 1, therefore violating the constraint. But if in that scenario we would instead have $\Delta(e_1) = 2$ and we hadn't had assumed that all the unscheduled generation -6 happens at the beginning, we would have falsely deduced that the given variable assignment could never be made resource consistent.

TRN limitations - Going Beyond Fixed Schedules
Notice that CP algorithm does not require the schedule to be fixed. For example, we could consider ATN to be $STNU$ and TC to be dynamic controllability ([Vidal and Ghallab, 1996]). The output is then an execution strategy, rather than a schedule. Notice that there is an important limitation to that approach though. Even though temporal schedule is dynamic, the schedule implied by resource constraints is static - we cannot change σ dynamically during execution.

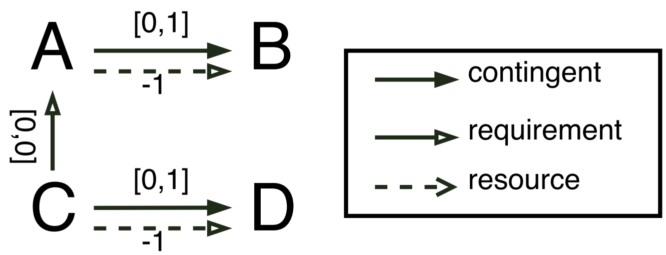

Figure 2: TRN cannot select σ dynamically. Number over a resource constraint arrow represents r in a simple resource constraint.

Figure 2 shows an example where TRN would report no solution found. However, if we ignore the resource constraints and find a dynamic execution strategy satisfying temporal constraints, it never violate resource constraints, as they are both generating. The reason TRN fails to find the solution is due to the fact that B and D are both in the set RE and TRN's solution attempts to fix the ordering between B and D, which is impossible to do statically in this example.

5 Experiments

5.1 TRN over STN

To understand the performance of our novel CP algorithm, we used the proposed MIP approach as a baseline. Both algorithms were used to determine time-resource consistency for TRN over Simple Temporal Network. In case of MIP based algorithm all the temporal constraints $l \leq x - y \leq u$, where $l, b \in \mathbb{R}$ and $x, y \in events(ATN)$ can be expressed as linear constraints, with x and y being continuous variables. In case of CP algorithm, we used Floyd-Warshall to determine temporal consistency as suggested in [Dechter *et al.*, 1991]. The test cases were created by the following procedure:

1. Specify number of events $N \geq 2$, number of temporal constraints $T \geq 2$ and number of resource constraints $R \geq 2$
2. Create a random schedule s for events in N with times in the interval $(0.0, 1.0)$.
3. Create T time constraints using the following procedure:
   (a) Choose start and end points $x, y \in N$.
   (b) Choose a type of constraint - lower bound or upper bound, each with probability 0.5
   (c) Let $d = s(y) - s(x)$ and chose number d' form exponential distribution with $\lambda = 1/\sqrt{d}$. For lower-bound set $l = d - d'$. For upper bound set $u = d + d'$.

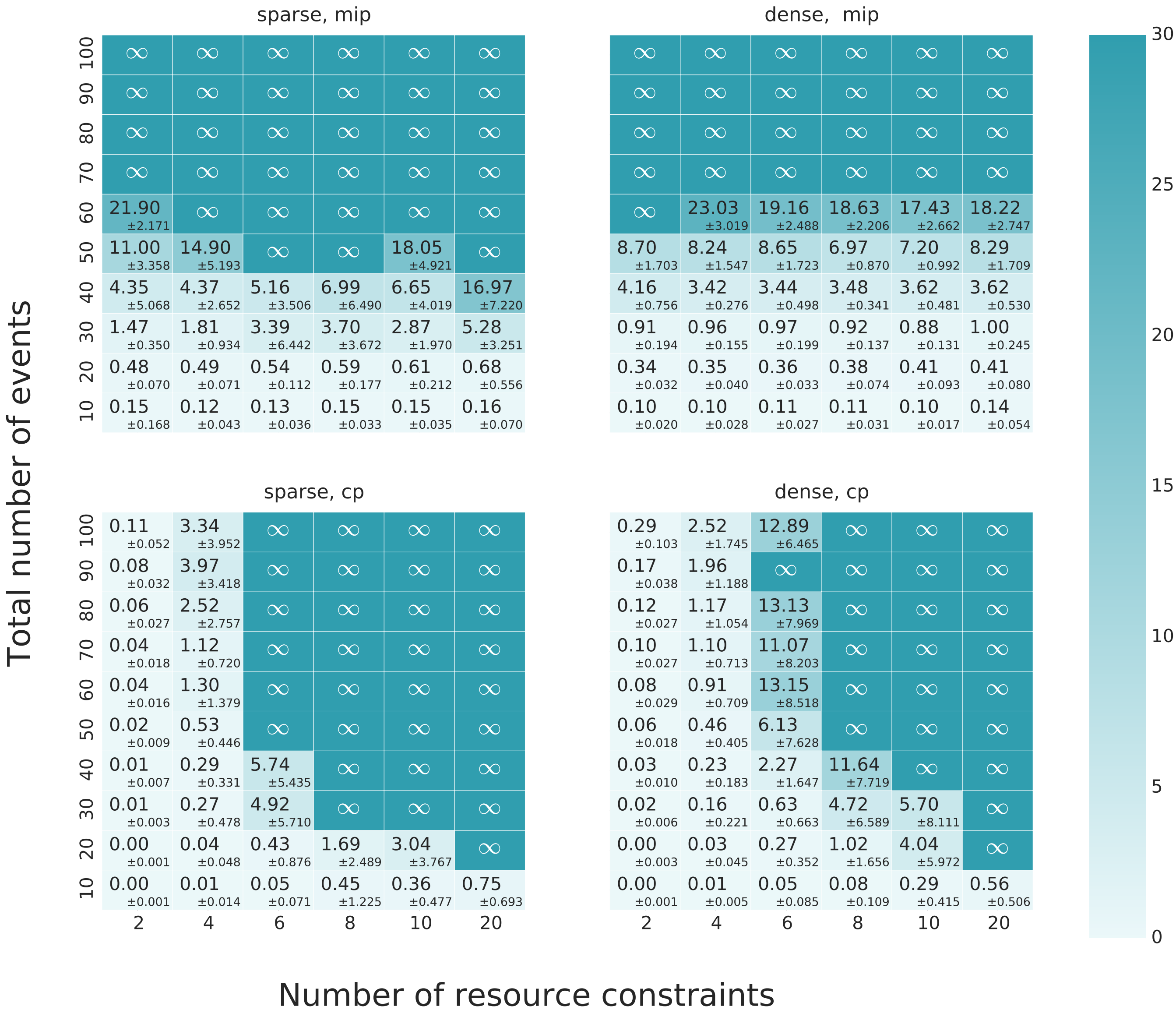

Figure 3: Comporison of execution time for different types of networks, or `inf` if the solver failed to compute the result within time limit. Y axis represents the number of events in the temporal network (N). X axis represents the number of resource constraints (R). Top portion of the figure was obtained using the MIP-based solver, while bottom part of the figure was obtained using CP-based solver. The left side of the figure represents computations on *sparse* networks, which in this case means that the total number of temporal constraints is $2N$. On the right side we have *dense* networks, meaning that the number of temporal constraints is $N^2/2$. This figure was computed by running the experiment for every set of parameters (but with different randomly generated instance) multiple times. Numbers in bottom left corner of each cell are corresponding standard deviations.

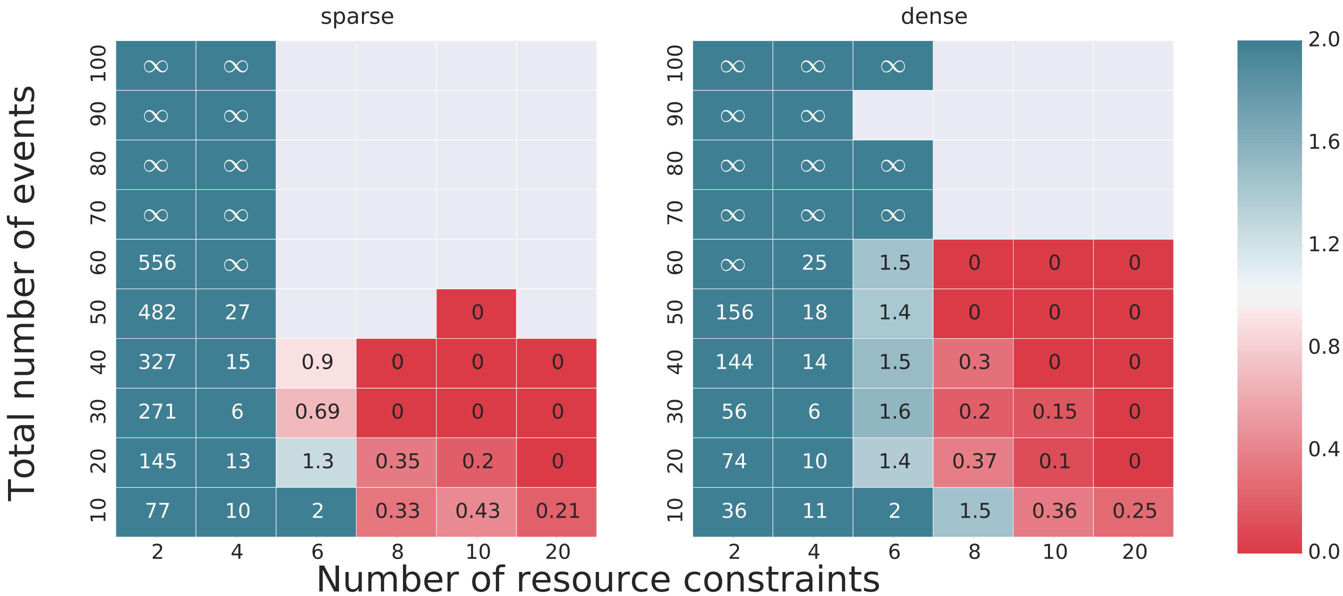

Figure 4: Number on the figure represents execution time using MIP-based algorithm divided by execution time using CP-based algorithm. Notice that in particular 0, means that CP-based algorithm failed to compute the results within the time limit and ∞ means that MIP-based algorithm timed out. The missing cells correspond to the networks where both of the algorithms timed out and therefore their execution time cannot be compared.

4. Choose number of generating constraints G as a random integer between 1 and $R-1$ and set number of consuming constraints as $C = R - G$ (so that there's at least on constraint of each type).
5. Create G generating constraints using the following procedure, by randomly choosing $x, y \in N$ and setting r to a random number between -1 and 0.
6. Create C consuming constraints using the following procedure.
   (a) Choose start and end points $x, y \in N$.
   (b) Let m be the maximum resource usage value between x and y considering all the resource constraints generated so far. If $m = 0$ repeat the process.
   (c) choose r from uniform distribution between 0 and $-m$.

We considered 10 different values of N: $10, 20, ..., 100$. We considered 6 different values of R: $2, 4, 6, 8, 10, 20$. We defined two types of networks - sparse, where $T = 2N$ and dense where $T = N^2/2$. For every set of parameters we run 15 trials. We set the time limit to 30 seconds. The results are presented on figure 3. We can see there exists a set of parameters where only CP managed to find the solution MIP exceed the time limit and vice versa. Figure 4 compares execution time of CP and MIP algorithms. The cells colored in blue are the ones where CP algorithm is faster and the cells colored in red are the ones where MIP based algorithm is better. One can see that CP is much better suited for large temporal networks with small number of resource constraints, while MIP scales much better with the number of resource constraints.

5.2 TRN over pSTN

To demonstrate extensibility of our approach we have implemented a version of TRN network, where the underlying temporal network is a pSTN ([Fang *et al.*, 2014]). pSTN extends the notion of STN. For this discussion we define STN events and edges as **actiavated time points** and **free constraints** respectively. pSTN defines **received time point** which is determined by the environment. Every received time point is defined by corresponding **uncertain duration (uDn)** constraint, which specifies a probability distribution over duration between an activated time point and a received time point. Due to that extension, the notion of consistency becomes probabilistic; rather than asking *is this pSTN consistent?*, we ask is *is this pSTN consistent with probability p?*. Since pSTN is an extension of STN, it is an ATN. Given the choice of p we can use probabilistic consistency as TC. Therefore we can use CP algorithm to check networks consistency. Example application of the algorithm and the schedule obtained is presented in the introduction.

6 Conclusion

In this paper, we have introduced Time Resource Networks, which allow one to encode many resource-constrained scheduling problems. We defined them in a way that permits use of many different notions of temporal networks to constrain schedules. We introduced a novel CP algorithm for determining time-resource consistency of a TRN and we compared it MIP baseline. We have demonstrated that our algorithm achieves superior performance for networks with large number of temporal constraints and small number of resource constraints. In addition, we have shown that CP algorithm is flexible and can support recently introduced probabilistic simple temporal networks [Fang *et al.*, 2014].

References

[Bartusch *et al.*, 1988] Martin Bartusch, Rolf H Möhring, and Franz J Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of operations Research*, 16(1):199–240, 1988.

[Bellman, 1956] Richard Bellman. Mathematical aspects of scheduling theory. *Journal of the Society for Industrial and Applied Mathematics*, 4(3):168–205, 1956.

[Dechter *et al.*, 1991] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

[Fang *et al.*, 2014] Cheng Fang, Peng Yu, and Brian C. Williams. Chance-constrained probabilistic simple temporal problems. In *AAA-14*, 2014.

[Garey *et al.*, 1976] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[Heimerl and Kolisch, 2010] Christian Heimerl and Rainer Kolisch. Scheduling and staffing multiple projects with a multi-skilled workforce. *OR spectrum*, 32(2):343–368, 2010.

[Johnson, 1954] Selmer Martin Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68, 1954.

[Markowitz and Manne, 1957] Harry M Markowitz and Alan S Manne. On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society*, pages 84–110, 1957.

[Morris and Muscettola, 2005] Paul H Morris and Nicola Muscettola. Temporal dynamic controllability revisited. In *AAAI*, pages 1193–1198, 2005.

[Morris *et al.*, 2001] Paul Morris, Nicola Muscettola, Thierry Vidal, et al. Dynamic control of plans with temporal uncertainty. In *IJCAI*, volume 1, pages 494–502. Citeseer, 2001.

[Patterson, 1984] James H Patterson. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management science*, 30(7):854–867, 1984.

[Pritsker *et al.*, 1969] A Alan B Pritsker, Lawrence J Waiters, and Philip M Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, 16(1):93–108, 1969.

[Vidal and Ghallab, 1996] Thierry Vidal and Malik Ghallab. Dealing with uncertain durations in temporal constraint networks dedicated to planning'. In *ECAI*, pages 48–54. PITMAN, 1996.

[Wagner, 1959] Harvey M Wagner. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, 6(2):131–140, 1959.