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

TODO

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fashion can be found in [Johnson, 1954], although there's evidence that the problem was already considered in unpublished versions of [Bellman, 1956].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 Programming solution to the scheduling problem and noticed that it *"is a single model which encompasses a wide variety of machine-scheduling situations"*.

In [Pritsker *et al.*, 1969] a generalization of scheduling problem is considered, which allows for multiple resource constraints. However the solution provided uses a discrete time formulation, which depending on required accuracy can substantially decrease performance. Work in this publication considers work on Temporal Networks, which explicitly model continuous time constraints. Interestingly, one of the publications about resource constrained scheduling ([Bartusch *et al.*, 1988]) used the notion of which can be thought of as resource constrained scheduling over Simple Temporal Networks (STN). The publication derives the theory behind STNs 3 years before the STN publication!

In [Dechter *et al.*, 1991] a notion of Simple Temporal Problem was introduced which allows one to solve problem with simple temporal constraints of form $l \leq t_y - t_x \leq u$. This simple concept was later extended with various more sophisticated notions of temporal constraints. [Vidal and Ghallab, 1996] defined the notion of 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); consistency of such temporal networks is called Strong Controllability. [Morris *et al.*, 2001] desribes a pseudopolynamial algorithm for handling uncertain duration constraint, where we are allowed to make a descition scheduling decitions based on knowledge of uncertain durtions from the past (Dynamic controllability). His algorithm is later improved to polynamial 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 abritrary probabilistic distribution.

3 Problem statement

In this section we will define notion of a Time Resource Network (TRN) and the relevant constraint on TRN's schedule - Resource Consistency. 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, for simplicity we only consider the problem of consistency, but the techniques presented in this paper can be easily extended to objective optimization over constrained schedules.

3.1 Abstract Temporal Network

Since TRNs can operate on top of many different types of temporal networks, we define a notion of Abstract Temporal Network (ATN), to capture only the necessary properties. For abstract temporal network we define two pieces of functionality:

1. `nodes(ATN)`, which returns a set of timepoints in ATN
2. `extend(ATN,` $\{stc_1, ...stc_n\}$`)`, which takes ATN and a set of simple temporal constraints ([Dechter *et al.*, 1991]) spanning `nodes(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 the obeying set of simple temporal constraint $\{stc_1, ...stc_n\}$. TC is a notion of probabilistic temporal consistency described in section 3.3.

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

3.2 Schedule

A schedule s : `nodes(ATN)` $\rightarrow \mathbb{R}$ is a mapping from abstract time points in ATN to concrete 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 that $TC(ATN)$ which means that there exists at schedule s such that $TC_s(ATN)$.

Example Example network that satisfies the ATN interface is Simple Temporal Network with Uncertainty (STNU) described in [Vidal and Ghallab, 1996]. Let N be an STNU. Using the terminology from the paper `nodes(`N`)` is the set of received and activated nodes and `extend(`N`,` $\{stc_1, ...stc_n\}$`)` augments N with stc_i encoded as a controllable link. One way to define is $TC(N)$ is to be true if and only if N is strongly controllable.

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 $x, y \in$ `nodes(ATN)` and $r \in \mathbb{R}$ is resource usage which can be positive (consumption) and negative (generation). Given a schedule s for any time $t \in \mathbb{R}$ resource **usage** for that simple resource constraint $src = (x, y, r)$ is equal to

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

Intuitively, simple resource constraint encodes the fact that between time $s(x)$ and $s(y)$ resource is consumed (generated) at the rate $|r|$ units of resource per unit time for positive (negative) r.

Our notation is inspired by [Bartusch *et al.*, 1988]. In particular notice that one can encode arbitrary piecewise-constant resource profile, by decomposing it into simple resource constraint expressing usages at every constant intervals and simple temporal constraints joining their ends (details can be found in [Bartusch *et al.*, 1988]).

3.5 Resource consistency

For a schedule s we define a **net-usage** at time $t \in \mathbb{R}$ as $U_s(t)$ in the following way:

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

Where R is a set of all the resource constraints and. 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 which is greater than the generation rate. Set C is introduced to make it easier to prove some 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 finite interval where $U_s > 0$. We say that TRN is resource consistent is there exists s such that $RC_s(TRN)$.

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 tackled 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 practically-resource-consistent if and only if*

$$\forall_{t \in nodes\ (ATN)} \lim_{\epsilon \to 0} U_s(s(t) + \epsilon) \leq 0 \tag{2}$$

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

Proof. $\Rightarrow$ Trivial from definition of resource-consistency. $\Leftarrow$ We say a time $t \in \mathbb{R}$ is scheduled if there exists a timepoint $x \in$ `nodes(ATN)` such that $t = s(x)$. Assume 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 $U_s(s(t_{danger})) > 0$. We will only consider the case where t_{danger} is **not** scheduled (because there are finitely many scheduled timepoints, we can consider them members of C). Let t_{before} be the highest scheduled timepoint that is smaller than t_{danger}. Notice that if no such timepoint exist, this means that there's no resource constraint (x, y, r) such that $s(x) \leq t_{danger} < s(y)$, so $U_s(s(t_{danger})) = 0$. 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 $U_s(s(t_{danger})) = \lim_{\epsilon \to 0} U_s(s(t_{before}) + \epsilon) > 0$. Contradiction.

□

Corollary 3.1.1. *Given a TRN with only simple resource constraints and two schedules A and B that have the same ordering of timepoints, A is p-resource-consistent if and only if B is p-resource-consistent.*

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

4 Approach

In this section be present two alternative approaches to solving the problem. One of them is using Mixed Integer Programming (MIP) and the other is using Constraint Satisfaction Problem (CSP) 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 timepoints relevant for resource constraints as $RT \subseteq nodes(ATN)$, i.e.

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

Additionally, let's introduce resource-change at timepoint $n \in nodes(ATN)$ as:

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

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. 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 propose the following formulation:

$$\forall_{t \in nodes(ATN)}. \quad 0 \leq t \leq M \quad (3)$$

$$\forall_{t_1, t_2 \in RT, t_1 \neq t_2}. \quad t_1 - t_2 \geq -x_{t_1, t_2} M \quad (4)$$

$$\forall_{t_1, t_2 \in RT, t_1 \neq t_2}. \quad t_1 - t_2 \leq (1.0 - x_{t_1, t_2}) M \quad (5)$$

$$\forall_{t_1, t_2 \in RT, t_1 \neq t_2}. \quad x_{t_1, t_2} + x_{t_2, t_1} = 1 \quad (6)$$

$$\forall_{t_1, t_2 \in RT, t_1 \neq t_2}. \quad x_{t_1, t_2} \in \{0, 1\} \quad (7)$$

$$\forall_{t_1 \in RT}. \quad \sum_{t_2 \in RT} x_{t_2, t_1} \Delta(t_2) \leq 0 \quad (8)$$

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

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_{t_1, t_2} are order variables, i.e.

$$x_{t_1, t_2} = \begin{cases} 1 & \text{if } s(t_1) \leq s(t_2) \\ 0 & \text{otherwise} \end{cases}$$

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 runtime. 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 $t \in nodes(ATN)$.

4.2 Constraint Satisfaction Programming based algorithm

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 timepoints. 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 $encode\_as\_scts$ 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 $1 \leftarrow 3$.

Implementation of `resource_consistent` follows from lemma 3.1 and is fairly straightforward - we can evaluate $\lim_{\epsilon \to 0} U_s(s(t) + \epsilon)$ for all the scheduled timepoints 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. Let's consider $RT = t_1, ..., t_N$. We define a problem using N variables: $x_1, x_2, ..., x_N \in \{1, ..., N\}$, such that $x_j = i$ if t_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 CSP equaivalent to algorithm 1:

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

$N \leftarrow$ `nodes(ATN);`
for $\sigma \leftarrow$ *permutation of* N **do**
  if *resource_consistent(R, σ)* **then**
    if *TC(extend(ATN, encode_as_scts(σ)))* **then**
      succeed;
    end
  end
end
fail;

Algorithm 1: Checking p-time-resource-consistency of a TRN

- **all_different_constraint** - ensure that all variables are different, i.e. the actually represent the permutation. This is standard constraint available in most CSP software packages.
- **time_consistent** - making sure that the temporal constraints implied by the permutation are not making the ATN inconsistent. Even if the variables are partially instantiated we can compute the all the temporal constraints implied by the permutation. For example if we only know that $x_1 = 3$, $x_5 = 2$ and $x_6 = 5$, that implies $t_5 \leq t_1 \leq t_6$.
- **resource_consistent** - ensure that for all $t_1, ..., t_n$, resource usage just after t_i is non-positive. Even if the order is partially specified we can still evaluate it. The tricky part is we need to assume that all the timepoints for which x_i is undergined and which are generating ($\delta(t_i) < 0$) could be scheduled before all the points for which order is defined. For exmaple if $N = 4$ and $\Delta(t_1) = 4$, $\Delta(t_2) = -6$, $\Delta(t_3) = 3$, $\Delta(t_4) = 4$ and we only know that $x_1 = 3$, $x_3 = 2$. The we have to assume that all the generation happened before the points that we know, i.e. initially resource usage is -6, then after t_3 is is -3, and after t_1 it is 1, therefore violating the constraint. But if in that scenario we would instead have $\Delta(t_1) = 2$ 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.

Going beyond schedules

Notice that the notion of the schedule is not explicitly used in CSP based algorithm. This means that in principle we are not required to use a static schedule, but we could for example consider ATN to be $STNU$ and TC to be dynamic controllability ([Vidal and Ghallab, 1996]). The output is then execution strategy, rather than a schedule. Notice that there's an important limitation to that approach though - some concepts might not naturally extend to resource constraints. For example in case of dynamic controllability, even though temporal schedule is dynamic, the schedule implied by resource constraints is static - we cannot changed σ dynamically during execution.

5 Experiments

5.1 TRN over STN

To compare both algorithms we chose a simple example of TRN over STN. In case of MIP based algorithm all the temporal constraints $l \leq x - y \leq u$, where $l, b \in \mathbb{R}$ and $x, y \in nodes(ATN)$ can be expressed as simple linear constraints, with x and y being continuous variables. In case of CSP based 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 nodes $N \geq 2$, number of temporal constraints $T \geq 2$ and number of resource constraints $R \geq 2$
2. Create a random schedule s for nodes 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 $u = d + d'$.
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: $2, 4, 6, 8, 10, 20$. To chose T 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 1. We can see there exist set of parameters where only CSP managed to find the solution MIP exceed the time limit and vice versa. Figure 2 compares execution time of CP and MIP algorithms. The cells colored in blue are the ones where CSP algorithm is faster and the cells colored in red are the ones where MIP based algorithm is better. One can see that CSP is much better suited for large temporal networks with small number of resource

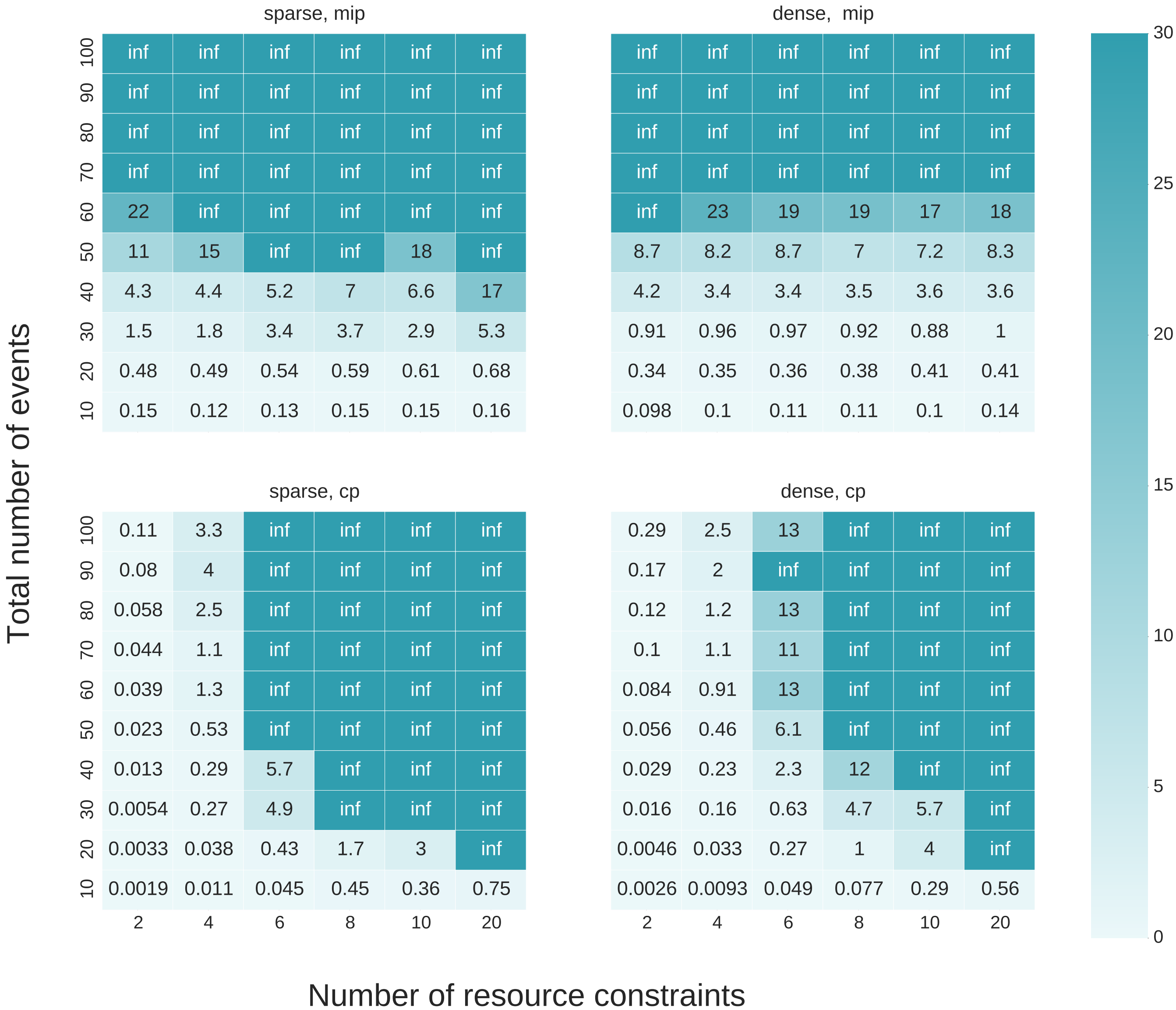

Figure 1: Comporison of execution time for different types of networks, or `inf` if the solver failed to compute the result within time limit. Y axis represents the number of nodes in the temporal network (N). X axis represents the number of resource constraints (R). Top portion of the figure was obtained using the MIP-based solver, while bottom part of the figure was obtained using CSP-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 $N^2/2$.

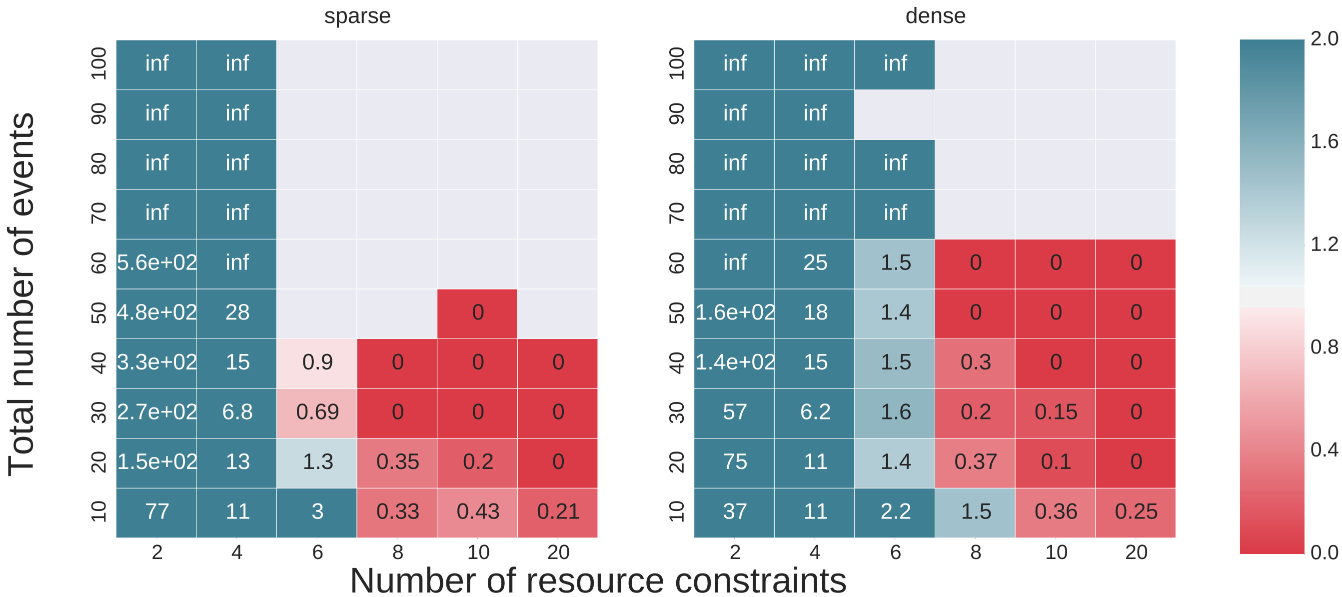

Figure 2: Number on the figure represents execution time using MIP-based algorithm divided by execution time using CSP-based algorithm. Notice that in particular 0, means that CSP-based algorithm failed to compute the results within the time limit and `inf` means that MIP-based algorithm timed out. The missing cells correspond to the networks where both of the algorithms timed out and therefore their execution time cannot be compared.

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 pSTN ([Fang *et al.*, 2014]). pSTN extends the notion of STN. For this discussion we define STN nodes 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.

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.

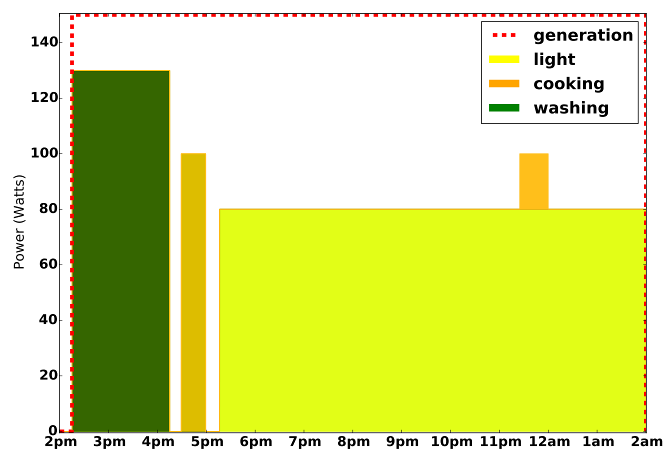

Figure 3: Depiction of solution to TRN spanning a pSTN.

Our algorithm successfully finds a solution to this scenario which meets the constraints with probability $99,7\%$, which is more than required. It is presented on fig. 3.

6 Future Work

linear resource constraint is a triplet (x, y, r_b, r_e), where $x, y \in \texttt{nodes(ATN)}$ and resource usage at time $s(x) \leq t \leq s(y)$ is equal to

$$u(t) = r_b + t\frac{r_e - r_b}{s(y) - s(x)}$$

Intuitively, simple resource constraint encodes the fact that between time $s(x)$ and $s(y)$ resource is consumed/generated with rate that changes linearly between $s(x)$ and $s(y)$.

probabilistic simple resource constraint Is an extension of simple resource constraint where r is a random variable (and therefore so is $u(t)$).

7 Conclusion

We have introduced a notion of Time Resource Network which is allows one to encode many resource-constrained scheduling problems. We defined it in a way that allows one to use arbitrary notion of temporal network to constrain schedules. We proposed two algorithms for determining time-resource consistency of a TRN and we have shown their strengths and weaknesses. We have demonstrated that our algorithm works for recently introduced probabilistic simple temporal networks.

Appendix A

Figure 1 was computed by running the experiment for every set of parameters multiple times. Figure 4 shows the corresponding standard deviations that can be helpful when judging relevance of results.

A References

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.

[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.

[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.

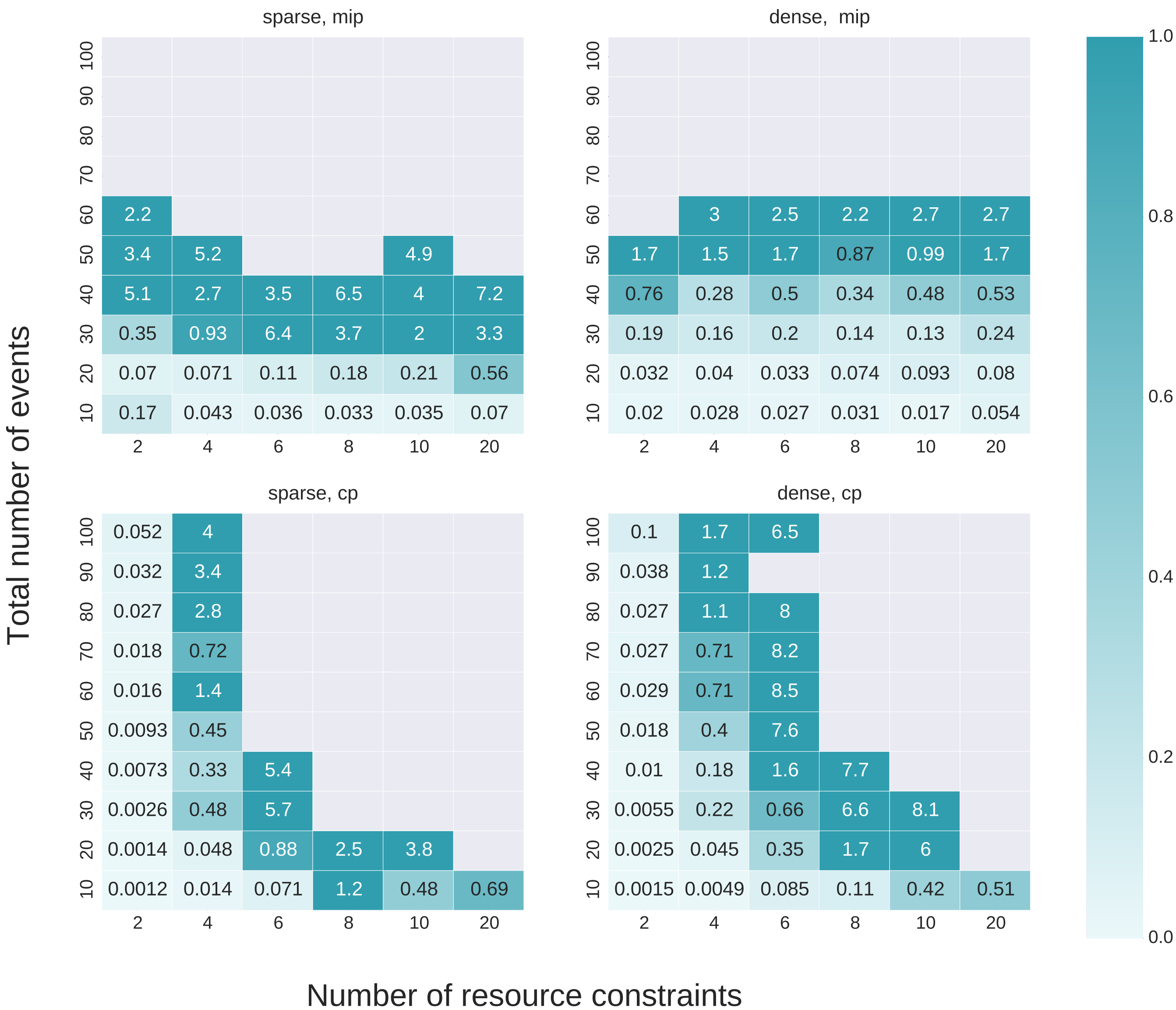

Figure 4: A standard deviation of results from figure 1. They are laid out in the same way as on that figure.