An incremental MaxSAT formulation for on-line train scheduling

bjørnar

Note for Carlo 2021-11-09

Re-scheduling trains after traffic disturbances is an important problem from the operations research literature.

There are several formulations of on-line train scheduling problem as mixed-integer linear programming (MILP), including time-indexed[**?**], big-M[**?**], path-and-cycle[**?**], and dynamic discretization discovery[**?**]. The constraints for the problem are disjunctions of difference constraints over a continuous time domain, though several of the MILP formulations are based on discretizing the time domain to create a problem over purely binary variables. However, since the optimization objective is usually formulated as a continuous piecewise-linear function, and it plays a crucial role in efficient train scheduling, most algorithmic efforts have focused on classical optimization techniques involving mixed-integer programming solvers (such as CPLEX and Gurobi) or local search algorithms.

The recent dynamic discretization discovery (DDD) formulation avoids altogether continuous variables for representing the arrival and departure times of trains. Instead, it solves an independent set problem, where the nodes consist of a sequence of consecutive time intervals for each time variable, and where edges are intervals that are incompatible under the difference constraints. The algorithm lazily subdivides the intervals and adds conflict edges between intervals until the difference constraints are satisfied by the left-most point of each interval. Because the objective is increasing in all time variables, selecting the earliest time from each interval is a lower bound on the objective value, and thus the algorithm is exact.

We adapt this algorithm for a SAT solver, using the same idea of abstraction refinement, though on unary-number-like representations of time-domain variables. The objective is discretized into a step-wise function (which has also, independently, been suggested by domain experts as a correct objective), so that the resulting problem becomes an incremental unweighted MaxSAT problem. We solve this problem using fully lazy constraint generation, in combination with the RC2 MaxSAT algorithm.

Potential problems with this approach

- The objective function does not change value at all when increasing the delay of a train above the maximum value. This allows the algorithm to effectively give up on optimizing a train that is sufficiently late, and delay it arbitrarily long. This seems to make the problem easier to solve in some cases, though the solution can be undesirable. For example, a train might be told to wait for hours in a station to keep the other trains on time.
- There are some inefficient runs where visits with maximum cost can be involved in very many conflicts (two trains can push each other forward in time indefinitely). This depends on the phase heuristic of the solver, and should be fixable.
- The performance is not directly comparable to the MILP approaches because of the change in objective function. The objective function from the instance files are ignored.
- The delays are minimized relative to the earliest possible arrival times, taking the delay into account, but not taking into account which trains are more delayed at the current time. Also, any priority between trains (train types) is ignored.

Problem definition

We simplify the on-line train scheduling problem using the following assumptions:

- Trains may exclusively occupy single-track sections.
- There is no rerouting.
- There are no capacity constraints in stations, i.e. any train can use any track and/or platform, and there are no constraints on how many trains can be in the same station at the same time.
- There are no safety margin on occupation times, i.e. a train can immediately enter a single-track section as soon as the previous train has left, and multiple trains can enter a station simultaneously.
- Trains occupy only one resource at a time (trains have effectively zero length).
- Trains take zero time to travel across a station.
- Double tracks are treated as two oppositely directed single-track connections, used exclusively for single-directed traffic.

We define the *simplified on-line train scheduling* input data as:

- A set of resources R and a binary relation defining the conflicting resources $C \subseteq R \times R$.
- A set of trains $t \in T$, each consisting of a sequence of visits V_t^i for $i = 0, 1, \ldots$.
- The visits V_t^i are a three-tuple $V_t^i = (r_t^i, e_t^i, l_t^i)$, where
  - $r_t^i \in R$ is a resource,
  - $e_t^i \in \mathbb{R}$ is the earliest time the train can enter the resource, and

  - $l_t^i \in \mathbb{R}$ is the minimum travel time the train needs to traverse the resource.

The constraints are:

- Earliest time constraints: for each visit V_t^i,

$$x_t^i \geq e_t^i$$

- Travel time constraints: for two consecutive visits V_t^i, V_t^{i+1},

$$x_t^i + l_t^i \leq x_t^{i+1}$$

- Resource constraints: for any pair of visits V_a^i, V_b^j that use conflicting resources $(r_a^i, r_b^j) \in C$,

$$(x_a^{i+1} \leq x_b^j) \vee (x_b^{j+1} \leq x_a^i)$$

The objective function is

$$\sum_{t \in T} \sum_{V_t^i} \sigma(x_t^i - e_t^i),$$

where x_t^i is the chosen time for train t to start its visit i, and $\sigma(x)$ is:

- $\sigma(x) = 3$, if $x > 360$ seconds
- $\sigma(x) = 2$, if $x > 180$ seconds
- $\sigma(x) = 1$, if $x > 0$
- $\sigma(x) = 0$, otherwise

Encoding

Dynamically discretized number representation

We define a dynamically discretized number y with a lower bound $lb(y)$ and upper bound $ub(y)$, as an increasing sequence of values and corresponding Boolean literals, initially containing:

$$[(lb(y), \top), (ub(y), \bot)]$$

We define the evaluation of the number y in a propositional logic model M as the value corresponding with the last (right-most) element in the sequence which evaluates to true. In the initial sequence, the evaluation will always be $lb(y)$.

Whenever we need to add constraints involving expressions of the form $y \geq c$, we check if c is a value in the sequence. If it is, we can use the corresponding Boolean literal to represent $y \geq c$, and it's negation to represent $y < c$. If c is not in the sequence (and $lb(y) < c < ub(y)$), we can add it by creating a new Boolean variable λ_y^c and inserting it in the sequence:

$$\left[(lb(y),\top),\left(c,\lambda_y^c\right),(ub(y),\bot)\right]$$

We also insert clauses to enforce consistency: each variable of the sequence implies the previous. Here, this is the non-clause $\lambda_y^c \Rightarrow \top$, though in general for neighboring variables λ_y^f and λ_y^g, we have:

$$\lambda_y^g \Rightarrow \lambda_y^c, \quad \lambda_y^c \Rightarrow \lambda_y^f$$

Because new values with corresponding fresh variables are inserted into the sequence at the correct place to preserve the ordering, we always have $f < c < g$. Note that when values can be incrementally added in arbitrary order, all the clauses above are sufficient for consistency, and valid, though some clauses become redundant because the involved variables are no longer neighbors in the sequence.

This encoding is similar to the number representation known as the Unary encoding[**?**] in that it represents lower and upper bounds, and uses one variable per possible value, but this version does not require all values to be represented, and as such it is similar to the generalized totalizer encoding [**?**].

With this representation, we can create Boolean logic constraints involving $y < c$ and $y \geq c$ for any specific c.

Note that the number may be either continuous or integral, in fact any totally ordered domain would work. In practice, we use integers representing seconds in the online train scheduling algorithm described below.

Discretized train scheduling constraints

Now, we implement each of the train scheduling constraints using dynamically discretized number representations of each x_t^i from the online train scheduling problem.

- **Earliest time constraints**: if we let $lb(x_t^i) = e_t^i$ (and $ub(x_t^i) = \infty$), then each constraint $x_t^i \geq e_t^i$ becomes $\top$, i.e. the constraint is implicit.
- **Travel time constraint**: consider consecutive visits x_t^i, x_t^{i+1}. For every possible value c, we have the clause

$$x_t^i \geq c \Rightarrow x_t^{i+1} \geq c + l_t^i.$$

- **Resource occupation constraint**: consider conflicting visits x_a^i, x_b^j. For every possible value c_1 and every possible value c_2, we have the clause

$$\Big((x_a^{i+1} \geq c_1) \Rightarrow (x_b^j \geq c_1)\Big) \vee \Big((x_b^{j+1} \geq c_2) \Rightarrow (x_a^i \geq c_2)\Big)$$

Although these are an infinte number of constraints, we can check a propositional model M for violations of the original constraints, and add values c to the dynamically discretized numbers only as needed, and then also create the corresponding discretized constraints.

Algorithm

Algorithm 1: Incremental MaxSAT algorithm for the online train scheduling problem

Input : Visits $V_t^i = (r_t^i, e_t^i, l_t^i)$ and objective function $\sigma(x)$.
Output: A value x_t^i for each visit, fulfilling the train scheduling constraints and minimizing the objective function.

```
1  $\mathcal{S} \leftarrow$ new incremental MaxSAT solver instance
2  $\mathcal{O} \leftarrow \{V_t^i : [(l_t^i, \top), (\infty, \bot)]\}$
3  $\mathcal{C} \leftarrow \{V_t^i : [(0, \top)]\}$
4  while true do
5      $M \leftarrow \mathcal{P}$.solve()      (RC2 increases cost until SAT)
6      $\{x_t^i\} \leftarrow \{\mathcal{O}[V_t^i]$.evalDiscretizedNumber($M$)$\}$
7      foreach $x_t^i + l_t^i > x_t^{i+1}$ do
8          $\mathcal{O}[V_t^{i+1}]$.newValue($x_t^i + l_t^i$)
9          $\mathcal{P}$.addClause($\neg\mathcal{O}[V_t^i]$.geq($x_t^i$) $\vee$ $\mathcal{O}[V_t^{i+1}]$.geq($x_t^i + l_t^i$))
10     end
11     foreach $(x_a^{i+1} < x_b^j) \wedge (x_b^{j+1} < x_a^i)$ do
12         $\mathcal{O}[V_a^i]$.newValue($x_b^{j+1}$)
13         $\mathcal{O}[V_b^j]$.newValue($x_a^{i+1}$)
14         $\mathcal{P}$.addClause($\neg\mathcal{O}[V_b^{j+1}]$.geq($x_b^{j+1}$) $\vee$ $\mathcal{O}[V_a^i]$.geq($x_b^{j+1}$) $\vee$
             $\neg\mathcal{O}[V_a^{i+1}]$.geq($x_a^{i+1}$) $\vee$ $\mathcal{O}[V_b^j]$.geq($x_a^{i+1}$))
15     end
16     if no constraints were added then
17         return $x_t^i$
18     end
19     foreach new value $c$ added to $V_t^i$ do
20         while $\mathcal{C}[V_t^i]$.maxValue $< \sigma(c - e_t^i)$ do
21             $\mathcal{C}[V_t^i]$.extendUnary($\sigma(c - e_t^i)$)
22             $\mathcal{S}$.addSoft($\neg\mathcal{C}[V_t^i]$.geq($\mathcal{C}[V_t^i]$.maxValue))
23         end
24         $\mathcal{P}$.addClause($\neg\mathcal{O}[V_a^{i+1}]$.geq($x_a^{i+1}$) $\vee$ $\mathcal{C}[V_t^i]$.geq($\sigma(c - e_t^i)$))
25     end
26 end
```

Performance evaluation

The tables below show the performance of the algorithm on 20 instances supplied by Anna Livia Croella. Each table uses a different objective function $\sigma^{a,b,c}$, where $\sigma(x)$ is:

- $\sigma(x) = c$, if $x > 360$ seconds
- $\sigma(x) = b$, if $x > 180$ seconds
- $\sigma(x) = a$, if $x > 0$
- $\sigma(x) = 0$, otherwise

Note that Instance 8 varies heavily in running time with different objective functions, while the other instance seem to work the same.

The column headings are:

- Insntc: the instance number.
- Trains: the number of trains in the instance.
- Resources: the number of single-track sections in the instance.
- Avg.res.: the average number of single-track sectionss that trains need to traverse.
- Confl.pairs: the total number of pairs of visits that share a resource.
- Cost: the minimum of the objective function
- n_{SAT}: the number of SAT problems solved that were satisfiable (resulting in further discretizing the domain).
- n_{UNSAT}: the number of SAT problems solved that were unsatisfiable (resulting in further increasing the cost).
- Travel confl.: the number of travel time constraints added.
- Res. confl.: the number of resource conflict constraints adeded.
- Vars: the number of Boolean variables in the final SAT problem.
- Clauses: the number of clauses in the final SAT problem.
- Solve time (ms): the total running time of the solver.

Insntc	Trains	Resources	Avg.res.	Confl.pairs	Cost	n_{SAT}	n_{UNSAT}	Travel confl.	Res. confl.	Vars	Clauses	Solve time (ms)
1	29	33	20.93	5915	630	55	630	928	284	4536	6575	63.92
2	25	33	19.00	3735	494	45	494	693	202	3370	4716	35.23
3	17	33	21.06	2061	356	55	356	513	130	2409	3373	23.56
4	14	33	19.21	1175	214	41	214	301	70	1361	1794	9.52
5	12	33	19.25	876	180	59	180	266	62	1191	1663	9.85
6	8	33	18.62	307	110	47	110	150	32	706	969	4.32
7	8	33	12.62	126	106	55	106	184	42	786	1173	5.77
8	28	33	19.21	4877	660	87	660	1627	682	7580	13937	484.22
9	21	33	19.10	2616	396	60	396	779	238	3717	6221	55.78
10	17	33	18.18	1632	190	64	190	419	116	1703	2789	21.39
11	18	25	17.33	2022	328	37	328	418	96	1974	2559	17.53
12	6	25	12.33	87	54	33	54	68	28	350	453	2.79
13	5	25	16.80	136	70	33	70	106	34	467	653	2.07
14	20	25	15.95	2246	284	57	284	580	182	2394	3712	23.23
15	5	25	17.80	136	82	34	82	85	22	457	549	2.35
16	6	25	12.67	93	72	34	72	88	12	404	494	1.98
17	24	25	15.25	2865	344	58	344	703	192	2814	4433	30.85
18	5	25	13.40	72	66	54	66	103	22	466	628	2.82
19	7	25	13.57	150	140	76	140	401	156	1588	2881	17.46
20	8	25	11.38	130	62	36	62	89	30	413	551	2.22

Table 1: Performance evaluation for $\sigma^{1,2,3}$

Insntc	Trains	Resources	Avg.res.	Confl.pairs	Cost	n_{SAT}	n_{UNSAT}	Travel confl.	Res. confl.	Vars	Clauses	Solve time (ms)
1	29	33	20.93	5915	846	57	846	908	274	5739	8297	89.09
2	25	33	19.00	3735	686	56	686	664	186	4326	5905	55.88
3	17	33	21.06	2061	530	86	530	587	130	3610	5111	45.04
4	14	33	19.21	1175	308	73	308	407	80	2110	2944	19.01
5	12	33	19.25	876	246	60	246	259	62	1513	2011	12.39
6	8	33	18.62	307	156	61	156	165	34	944	1318	6.25
7	8	33	12.62	126	144	48	144	163	36	921	1299	5.39
8	28	33	19.21	4877	1008	86	1008	1505	696	13591	27179	12203.59
9	21	33	19.10	2616	620	85	620	947	286	6128	10528	137.02
10	17	33	18.18	1632	278	107	278	496	136	2744	4650	45.86
11	18	25	17.33	2022	510	62	510	483	118	3073	4199	33.87
12	6	25	12.33	87	74	39	74	70	26	441	566	2.20
13	5	25	16.80	136	106	30	106	100	34	614	829	2.74
14	20	25	15.95	2246	388	46	388	601	190	3760	6272	96.25
15	5	25	17.80	136	126	36	126	86	22	646	800	3.14
16	6	25	12.67	93	106	38	106	101	18	591	759	2.83
17	24	25	15.25	2865	498	52	498	697	194	4150	6544	50.66
18	5	25	13.40	72	90	59	90	106	24	622	841	3.17
19	7	25	13.57	150	220	61	220	353	148	2278	3958	21.85
20	8	25	11.38	130	90	51	90	111	30	589	799	3.30

Table 2: Performance evaluation for $\sigma^{1,3,6}$

Insntc	Trains	Resources	Avg.res.	Confl.pairs	Cost	n_{SAT}	n_{UNSAT}	Travel confl.	Res. confl.	Vars	Clauses	Solve time (ms)
1	29	33	20.93	5915	924	53	924	864	274	6189	8775	96.86
2	25	33	19.00	3735	746	60	746	663	192	4709	6364	64.07
3	17	33	21.06	2061	604	96	604	660	132	4474	6479	60.43
4	14	33	19.21	1175	342	88	342	501	84	2592	3796	25.53
5	12	33	19.25	876	280	95	280	299	64	1899	2627	18.76
6	8	33	18.62	307	156	59	156	155	34	936	1277	6.26
7	8	33	12.62	126	174	49	174	156	38	1054	1459	6.46
8	28	33	19.21	4877	1224	88	1224	1544	674	17522	36935	18202.19
9	21	33	19.10	2616	752	94	752	1027	284	7565	12734	167.48
10	17	33	18.18	1632	314	103	314	480	156	3052	5072	45.40
11	18	25	17.33	2022	580	65	580	491	120	3567	4874	44.25
12	6	25	12.33	87	86	39	86	69	26	512	661	3.01
13	5	25	16.80	136	142	30	142	99	34	758	1003	3.54
14	20	25	15.95	2246	436	41	436	610	204	4149	6791	69.28
15	5	25	17.80	136	168	35	168	88	22	833	1035	4.17
16	6	25	12.67	93	112	36	112	99	18	623	801	2.87
17	24	25	15.25	2865	590	46	590	707	206	5650	9818	218.14
18	5	25	13.40	72	96	54	96	100	22	640	854	3.09
19	7	25	13.57	150	280	64	280	391	152	3462	6146	45.78
20	8	25	11.38	130	108	74	108	141	30	762	1048	5.32

Table 3: Performance evaluation for $\sigma^{1,3,9}$