

# P<sup>3</sup>C: A New Algorithm for the Simple Temporal Problem\*

Léon Planken<sup>a</sup>      Roman van der Krogt<sup>b</sup>      Mathijs de Weerd<sup>a</sup>

<sup>a</sup> *Faculty EEMCS, Delft University of Technology, The Netherlands*

<sup>b</sup> *Cork Constraint Computation Centre, University College Cork, Ireland*

The Simple Temporal Problem (STP) was first proposed by Dechter et al. [1]. An STP instance consists of a set of time-point variables and a set of constraints that bound the time difference between pairs of these variables. The goal is then to find out all possible time values that can be assigned to these variables, or the message that no such assignment exists. The STP has received widespread attention that still lasts today, with applications in diverse areas. Moreover, the STP appears as a pivotal subproblem in more expressive formalisms for temporal reasoning, such as the Temporal Constraint Satisfaction Problem—proposed by Dechter et al. in conjunction with the STP.

An STP instance  $\mathcal{S}$  consists of a set  $V = \{v_1, \dots, v_n\}$  of time-point variables representing events, and a set  $E$  of  $m$  constraints over pairs of time points, bounding the time difference between events. Every constraint  $c_{i \rightarrow j}$  has a weight  $w_{i \rightarrow j} \in \mathbb{R}$  corresponding to an upper bound on the time difference, and thus represents an inequality  $v_j - v_i \leq w_{i \rightarrow j}$ . The STP can be conveniently represented as a constraint graph. A *solution* to the STP instance is an assignment of a real value to each time-point variable such that the differences between each constrained pair of variables fall within the range specified by the constraint. Note that many solutions may exist; to capture all of them, we are interested in calculating an equivalent *decomposable* STP instance, from which all solutions can then be extracted in a backtrack-free manner.

The fastest known algorithm to attain decomposability is  $\Delta$ STP [3], included as Algorithm 1. Instead of considering a complete constraint graph, as its predecessors do, this algorithm works on a *chordal* constraint graph, which generally contains far less edges. A graph is chordal if every cycle of length greater than 3 contains a *chord*, i.e. an edge connecting two nonadjacent vertices. Chordality can efficiently be enforced in  $\mathcal{O}(nm)$  time [2]. Further, it is known in graph theory that a graph is chordal if and only if its vertices can be ordered in what is called a *simplicial elimination ordering*. For details, we refer to our full paper; however, it should be noted that this ordering is a byproduct of enforcing chordality.

For our new algorithm P<sup>3</sup>C, included as Algorithm 2, we assume the vertices are ordered in this way. To enforce decomposability, the algorithm then performs just a forward and a backward sweep along the triangles in the graph, whereas  $\Delta$ STP considers each triangle at least once. By this consideration, P<sup>3</sup>C will never be slower than  $\Delta$ STP. Moreover, we constructed a class  $\mathcal{P}$  of pathological STP instances for which we can prove the following result:

**Theorem 1.** *When solving an instance from  $\mathcal{P}$  on  $t$  triangles, the  $\Delta$ STP algorithm may require processing  $\Omega(t^2)$  triangles.*

In contrast, P<sup>3</sup>C's runtime can be strictly bounded:

**Theorem 2.** *Algorithm P<sup>3</sup>C achieves decomposability in time  $\Theta(t)$ , where  $t$  is the number of triangles in the (chordal) STP. If the instance is inconsistent, this is discovered in time  $\mathcal{O}(t)$ .*

Upper bounds on the time complexity can also be expressed in other parameters:

**Corollary 3.** *The P<sup>3</sup>C algorithm has time complexity  $\mathcal{O}(n\delta^2) \subseteq \mathcal{O}(n^3)$ . Here,  $\delta$  is the graph degree (i.e. the maximum number of neighbours of a single vertex).*

We empirically validated these results with tests on the specially constructed pathological STP instances as well as more practical test cases. Some results are included in Figure 1.

---

\*Published in: Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), AAAI Press, 2008

```

Q ← all triangles in S
while ∃T ∈ Q do
  foreach permutation (vi, vj, vk) of T do
    wi→k ← min(wi→k, wi→j + wj→k)
    if wi→k has changed then
      if wi→k + wk→i < 0 then
        return INCONSISTENT
      end
      Q ← Q ∪ {all triangles  $\hat{T}$  in S |
        vi, vk ∈  $\hat{T}$ }
    end
  end
end
Q ← Q \ {T}
end

```

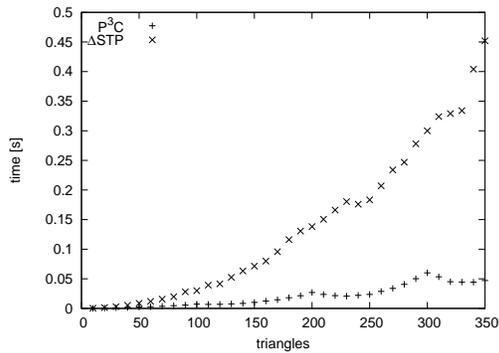
Algorithm 1:  $\Delta$ STP

```

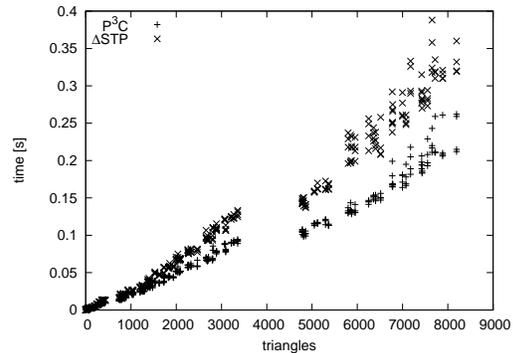
for k ← n to 1 do
  forall i, j < k such that
    {vi, vk}, {vj, vk} ∈ E do
      wi→j ← min(wi→j, wi→k + wk→j)
      if wi→j + wj→i < 0 then
        return INCONSISTENT
      end
    end
  end
end
for k ← 1 to n do
  forall i, j < k such that
    {vi, vk}, {vj, vk} ∈ E do
      wi→k ← min(wi→k, wi→j + wj→k)
      wk→j ← min(wk→j, wk→i + wi→j)
    end
  end
end

```

Algorithm 2: P<sup>3</sup>C



(a) Pathological instances



(b) Job-shop problem

Figure 1: Test results

## References

- [1] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3):61–95, 1991.
- [2] Uffe Kjærulff. Triangulation of graphs - algorithms giving small total state space. Technical report, Aalborg University, March 1990.
- [3] Lin Xu and Berthe Y. Choueiry. A new efficient algorithm for solving the Simple Temporal Problem. In *Proceedings of TIME-ICTL*, pages 210–220, Los Alamitos, CA, USA, 2003. IEEE Computer Society.