University of Nebraska - Lincoln
DigitalCommons@University of Nebraska - Lincoln

CSE Conference and Workshop Papers | Computer Science and Engineering, Department of

1-1-2004

# Evaluating Consistency Algorithms for Temporal Metric Constraints

Yang Shi
*University of Nebraska-Lincoln*, yshi@cse.unl.edu

Anagh Lal
*University of Nebraska - Lincoln*, alal@cse.unl.edu

Berthe Y. Choueiry
*University of Nebraska - Lincoln*, choueiry@cse.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/cseconfwork

Part of the Computer Sciences Commons

Shi, Yang; Lal, Anagh; and Choueiry, Berthe Y., "Evaluating Consistency Algorithms for Temporal Metric Constraints" (2004). *CSE Conference and Workshop Papers*. Paper 166.
http://digitalcommons.unl.edu/cseconfwork/166

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

# Evaluating Consistency Algorithms for Temporal Metric Constraints

Yang Shi, Anagh Lal, and Berthe Y. Choueiry
Constraint Systems Laboratory • Computer Science & Engineering • University of Nebraska-Lincoln • yshi, alal, choueiry@cse.unl.edu

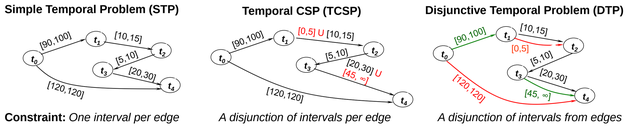
## Summary

**Focus:** Networks of temporal metric constraints
**Task:** Evaluating the performance of algorithms for
- Determining the consistency of the Simple Temporal Problem (STP)
- Finding the minimal network of the Temporal Constraint Satisfaction Problem (TCSP)

**Future:** Enhance triangulation-based algorithms with incrementality

## Networks of Temporal Metric Constraints

**Temporal constraint network:** a graph G=(V, E, I) where
- V: set of vertices representing time points $t_i$
- E: set of directed edges representing constraints between two time points $t_i$ & $t_j$
- I: set of constraint labels for the edges. A label is a set of intervals and an interval [a, b] denotes a constraint of bounded differences ($a \le t_j - t_i \le b$)




**Constraint:** *One interval per edge* — *A disjunction of intervals per edge* — *A disjunction of intervals from edges*

**Minimal network:** Make labels of binary constraints as tight as possible
**Solution:** Find a value for each variable satisfying all temporal constraints
**Consistency:** Determine whether a solution exists

| | STP | TCSP | DTP |
|---|---|---|---|
| **Minimal network** | P | NP-hard | NP-hard |
| **Consistency** | P | NP-complete | NP-complete |

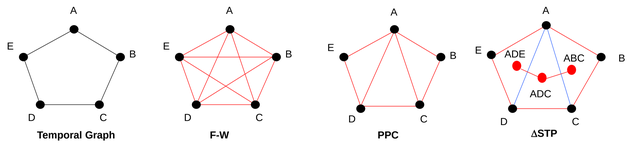
*Problem complexity*

## Algorithms for the STP

### Determining consistency
- Directional Path Consistency (DPC)
- Bellman-Ford (BF), single-source shortest paths
- Incremental version of Bellman-Ford (incBF) [Cesta & Oddi, TIME 96]

### Determining consistency & finding minimal network
- Floyd-Warshall (F-W), all-pairs shortest paths
- Partial Path Consistency (PPC) [Bliek & Haroud, IJCAI 99]
- ΔSTP: an improvement of PPC [Xu & Choueiry, TIME 03]

### Properties & advantages of ΔSTP
**ΔSTP** considers the temporal graph as composed of triangles instead of edges

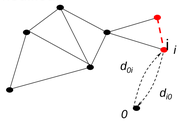



*Constraint graphs of different solving strategies*

1. A finer version of PPC.
2. Cheaper than PPC and F-W.
3. Guarantees the minimal network.
4. Automatically decomposes the graph into its bi-connected components:
   - binds effort in size of largest component.
   - allows parallellization.
5. Best known algorithm for computing the minimal network of an STP

### An incremental version of BF (incBF):
When adding a constraint, incBF visits only nodes whose distance to origin is modified:

1. Allows dynamic updates for both constraint posting & retraction.
2. Localizes effects of change.
3. Determines consistency of STP *by does not yield the minimal network.*
4. Can detect inconsistency much earlier than BF by detecting negative cycles ($d_{0i} + d_{i0} < 0$).
5. Is useful for TCSP: incrementality is useful for checking the consistency of STPs in the search tree of the meta-CSP.



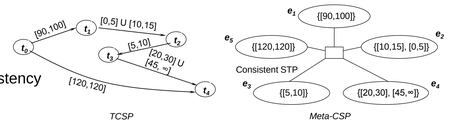

*Point 0 is the source added to the constraint graph. incBF updates only affected distances, and detects inconsistency when re-visiting a node.*

### Comparing the above strategies:

| | Graph | Complexity | Consistency | Minimality |
|---|---|---|---|---|
| **F-W** | Complete | $\Theta(n^3)$ | Yes | Yes |
| **DPC** | Triangulated | $O(nW^*(d)^2)$ very cheap | Yes | No |
| **PPC** | Triangulated | $O(n^3)$ Usually cheaper than F-W/PC | Yes | Yes |
| **ΔSTP** | Triangulated | Always cheaper than PPC | Yes | Yes |
| **BF/incBF** | Source point is added | $O(en)$ | Yes | No |

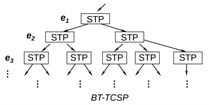
## Solving the TCSP [Dechter et al. AIJ 91]

**TCSP is formulated as a meta-CSP**
- Variables: edges of the constraint network
- Domains of variables: edge labels in the constraint network
- A unique global constraint: checking consistency of an STP




The minimal network of the TCSP can be found by computing *all the solutions* to the meta-CSP

When using backtrack search for finding all the solutions to the meta-CSP (BT-TCSP), every node in the search tree is an STP to be checked for consistency
→ An exponential number of STPs to be considered!

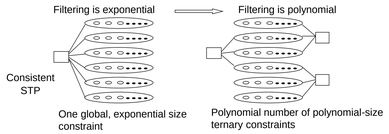



## Improving Search for the TCSP [Xu & Choueiry CP 03]

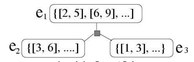
**Improve the performance of BT-TCSP:**
- **ΔAC:** a consistency filtering algorithm for reducing the size of TCSP.
- Exploit the topology of the constraint graph:
  - **AP**: using articulation points
  - **NewCyc**: a heuristic for avoiding unnecessary checking of STPs at every node.
  - **EdgeOrd**: a variable ordering heuristic.

### ΔAC: A new algorithm for filtering TCSP
**ΔAC** removes inconsistent intervals from the domain of the variables of the meta-CSP to reduce the size of meta-CSP:
- In a pre-processing step (implemented)
- In a look-ahead strategy (to be tested)

[Choueiry & Xu, AICom 04]



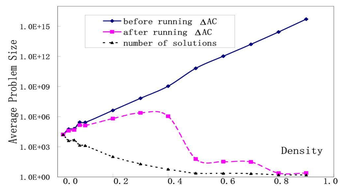

**ΔAC** checks combinations of 3 intervals:




✓ [2,5] composed with [1, 3] intersects with [3, 6]
✓ [1,3] composed with [3, 6] intersects with [2, 5]
✗ [3,6] composed with [2, 5] does not intersect with [1, 3]

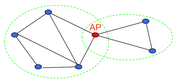
ΔAC removes [1, 3] from domain of $e_3$.

Advantages of **ΔAC:**
- It is effective, especially under high density.
- It is sound, cheap *O (n |E |k3),* may be optimal.
- It uncovers a phase transition in TCSP.



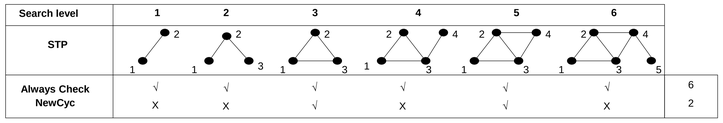

*Reduction of problem size of the TCSP*

### Articulation Points (AP) exploits the topology of the graph
- Decomposes the graph into bi-connected components.
- Solves each of them independently.
- Binds the total cost by the size of largest component.



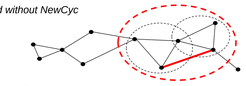

*Decomposition Using Articulation Point*

### New cycle check (NewCyc) eliminates unnecessary STP-consistency checks
- Checks presence of new cycles *O* (|*E* |).
- Checks consistency only when a new cycle is added.
- Does not affect number of nodes visited in BT-TCSP.




*Number of STP checks with and without NewCyc*

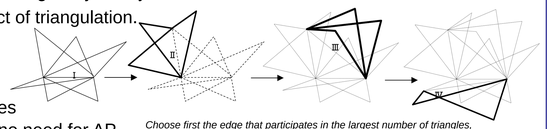
Advantages of **NewCyc**:
- Reduces effort of consistency checking.
- Restricts effort to new bi-connected component.



*Checks the consistency of only the newly formed biconnected component*

### Edge Ordering (EdgeOrd): a variable ordering heuristic in BT-TCSP
- Orders the edges using "triangle adjacency".
- Priority list is a by-product of triangulation.

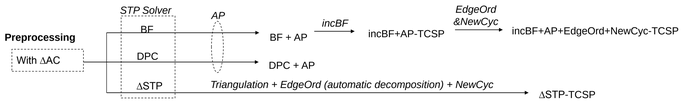
Advantages of **EdgeOrd**
- Localizes backtracking.
- Automatically decomposes the constraint graph ⇒ no need for AP.




*Choose first the edge that participates in the largest number of triangles, then consider in priority the edges of the triangles where it appears*

## Experiments

**We tested the following combinations:**

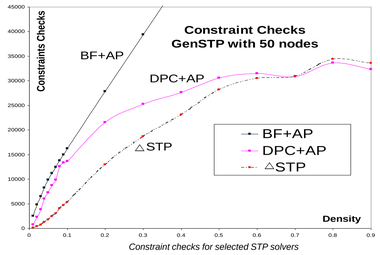



**Random generators of STP & TCSP:**
- Generators take as input:
  1. Number of time points of the TCSP
  2. Constraint density
  3. (Number of intervals per edge)
  4. Percentage of problems guaranteed consistent
- Note that size of meta-CSP is exponential in the number of time points

**Measured:**
CPU time, NV number of nodes visited (for TCSP), & CC number of constraint checks

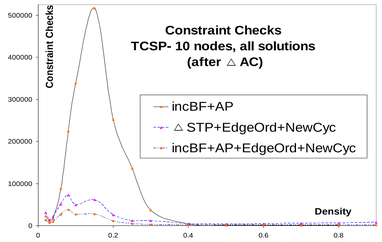
## Results of Empirical Evaluations

### Experiments on the STP:
- 50-node STP, density in [2%, 90%], 100 samples per point
- ΔSTP results in the minimal network & dominates all others
- Cost of BF increases linearly with density (bounded by *O(en)*, where *n* and *e* are respectively the number of nodes and the number of edges in the graph).



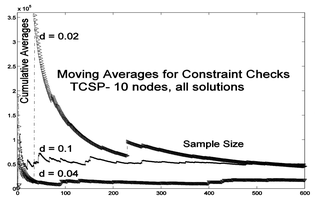

*Constraint checks for selected STP solvers*

### Experiments on TCSP (all solutions):
- 10-node TCSP, density in [2%, 90%], 600 samples per point
- Search enhanced with ΔAC, AP, NewCyc, EdgeOrd




*Constraint checks for selected TCSP solvers (finding minimal network of the TCSP)*

For small density values (<0.1), values of results were instable. We increased number of samples up to 600 samples per point:




*For small values of d, the average of CC, average of CC is not stable when the sample size is less than 400.*

| d | Δ STP $CCx10^3$ | incBF $CCx10^3$ | Gain $CCx10^3$ | | |
|---|---|---|---|---|---|
| | | | LL | Average | UL |
| 0.02 | 45.61 | 14.77 | 5.39 | 30.84 | 56.29 |
| 0.04 | 17.51 | 7.56 | 5.06 | 9.95 | 14.84 |
| 0.06 | 51.66 | 24.30 | 3.45 | 27.35 | 51.24 |
| 0.08 | 83.38 | 50.74 | 4.86 | 32.63 | 60.41 |
| 0.10 | 50.31 | 26.24 | 20.29 | 24.07 | 27.84 |
| 0.15 | 75.92 | 37.61 | 20.52 | 38.30 | 56.08 |
| 0.20 | 28.09 | 12.03 | 10.74 | 16.06 | 21.38 |

*Average CC gain of the best strategy and its lower limit (LL) and upper limit (UL) with 95% confidence.*

## Conclusions

**For STP:** ΔSTP outperforms all others
**For TCSP:**
- incBF outperforms ΔSTP
- EdgeOrd & NewCyc always beneficial

**Future: exploit incrementality**

| Algorithm | Performance Ranking | |
|---|---|---|
| | STP | TCSP |
| FW + AP | worse | worse |
| DPC + AP | better | OK |
| BF + AP | OK | - |
| Δ STP | best | - |
| incBF + AP | good | good |
| Δ STP + EdgeOrd + NewCyc | - | better |
| incBF + AP + EdgeOrd + NewCyc | - | best |

## References

Cesta & Oddi. Gaining Efficiency and Flexibility in the Simple Temporal Problem. TIME 96
Choueiry & Xu. An Efficient Consistency Algorithm for the Temporal Constraint Satisfaction Problem. AICom 2004.
Dechter, Meiri, & Pearl. Temporal Constraint Networks. AIJ 91.
Stergiou & Koubarakis. Backtracking Algorithms for Disjunctions of Temporal Constraints. AIJ 2000.
Xu & Choueiry. A New Efficient Algorithm for Solving the Simple Temporal Problem. TIME 2003.
Xu & Choueiry. Improving Backtrack Search for Solving the TCSP. CP 2003.

Support: Layman award, NASA-Nebraska grant, NSF CAREER Award #0133568

University of Nebraska Lincoln