Consistency Methods for Temporal Reasoning

Berthe Y. Choueiry
Constraint Systems Laboratory
Department of Computer Science & Engineering
University of Nebraska-Lincoln

Joint work with Lin Xu (currently Ph.D. student at UBC), Shi Yang, and Anagh Lal

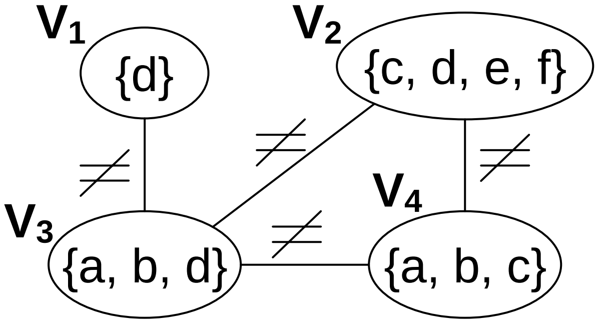
Supported by
- a NASA-Nebraska grant,
- CAREER Award #0133568, and
- a gift from Honeywell Laboratories

Outline

- Background and motivation
- Contributions
- Future work

One-slide introduction to CSPs

- **Problem definition**
  - Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$
    - $\mathcal{V}$: set of variables
    - $\mathcal{D}$: set of their domains
    - $\mathcal{C}$: set of constraints (relations) restricting the acceptable combination of values for variables
    - Solution is a consistent assignment of values to variables
  - Query: find 1 solution, all solutions, etc.



- **Solution techniques**
  - Constraint propagation (inference)
  - Search (enumeration)
- **Topic today**: temporal networks

Reasoning about time

- Tom wants to serve tea
  - Cleaning the teapot: 2 min
  - Cleaning the tea cups: 5 min
  - Boiling the water: 8 min
  - Preparing tea: 10 to 15 min

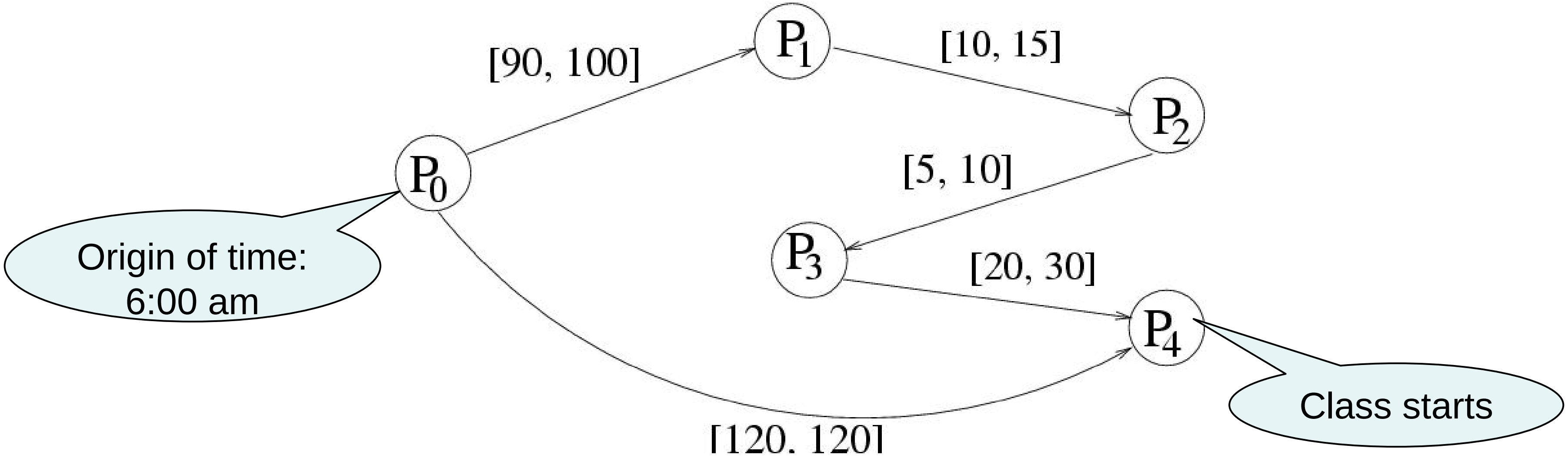
- Useful for..
  - Executing plans, understanding stories, solving crimes...

Temporal reasoning in AI

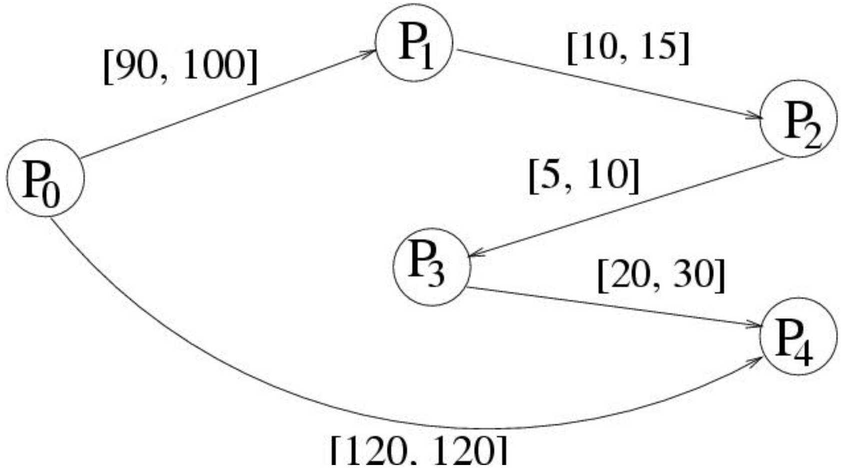
- Philosophies of time
- Temporal logic
- Temporal constraint networks: events and time points
  - Qualitative relations: Before, after, during, etc.
  - Quantitative (metric) relations: 10 min before, 15 min duration, etc.
    - ✓ Simple Temporal Problem (STP)
    - ✓ Temporal Constraint Satisfaction Problem (TCSP)
    - ✗ Disjunctive Temporal Problem (DTP)

Simple example

Tom has class at 8:00 a.m. Today, he gets up between 7:30 and 7:40 a.m. He prepares his breakfast (10-15 min). After breakfast (5-10 min), he drives to school (20-30 min). Will he be on time to class?



Simple Temporal Problem (STP)



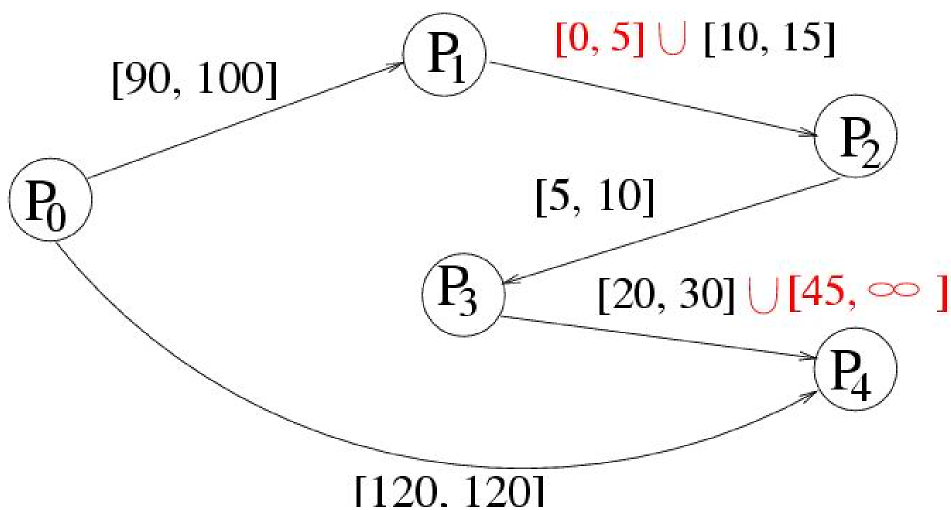
- **Variable**: Time point for an event
- **Domain**: A set of real numbers
- **Constraint**: distance between time points

$$[5, 10] \Leftrightarrow 5 \leq P_b - P_a \leq 10$$

- **Algorithm**: Floyd-Warshall, polynomial time

More complex example

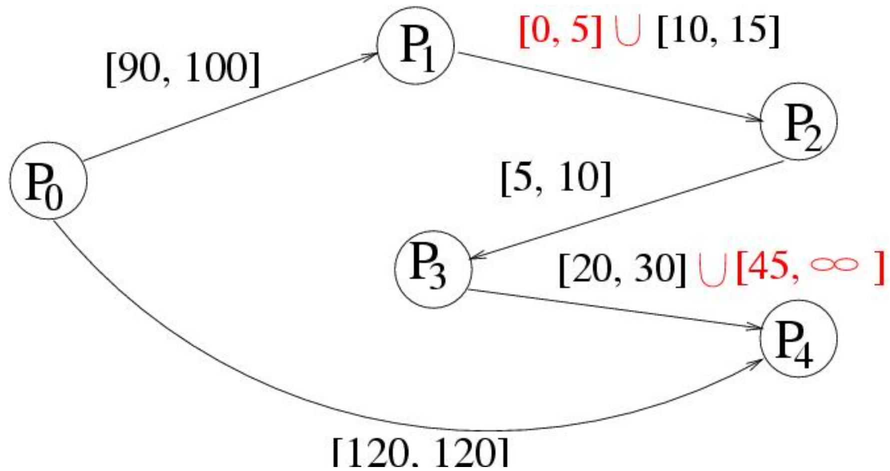
Tom has class at 8:00 a.m. Today, he gets up between 7:30 and 7:40 a.m. He either makes his breakfast himself (10-15 min), or gets something from a local store (less than 5 min). After breakfast (5-10 min), he drives to school (20-30 min) or takes the bus (at least 45 min).



Possible questions

- Can Tom arrive school in time for class?
- Is it possible for Tom to take the bus?
- If Tom wanted to save money by making breakfast for himself and taking the bus, when should he get up?

Temporal CSP (TCSP)

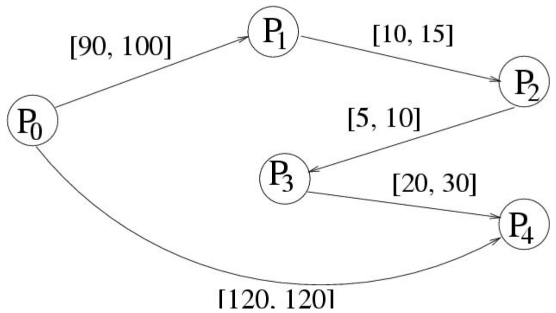


- **Variable, domain:** same as STP
- **Constraint**: a disjunction of intervals

  $[0, 5] \cup [10, 15]$

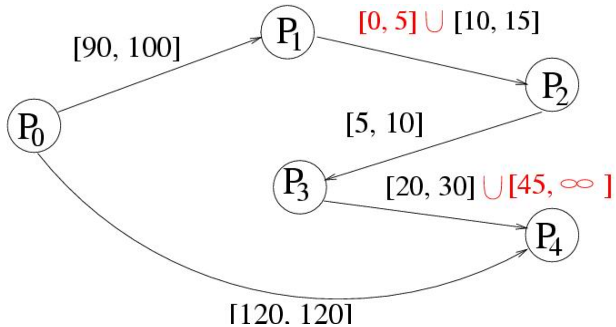
- **Algorithm**: Backtrack search [Dechter+, 91]

Temporal networks: STP ⊆ TCSP ⊆ DTP



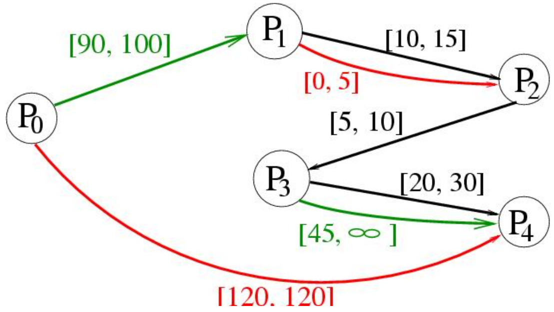
Simple Temporal Problem (STP)

- Each edge has a unique (convex) interval



Temporal CSP (TCSP)

- Each edge has a disjunction of intervals
- STP ⊆ TCSP [Dechter+, 91]

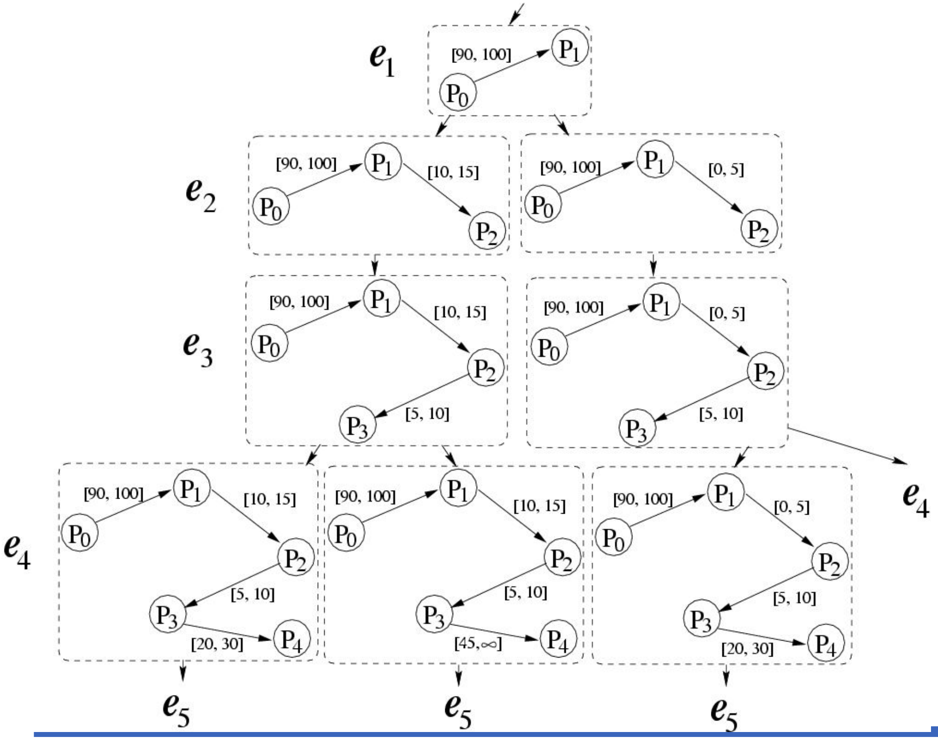


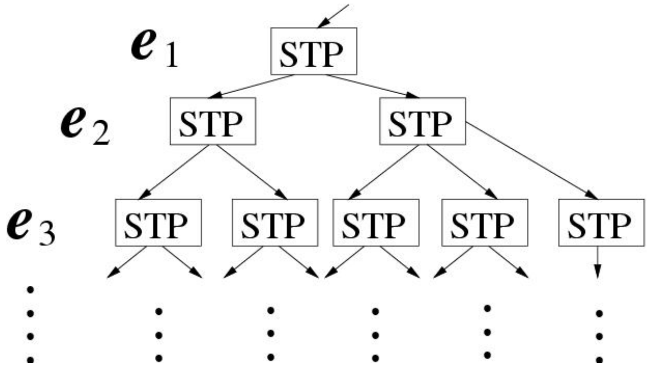
Disjunctive Temporal Problem (DTP)

- Each constraint is a disjunction of edges
- TCSP ⊆ DTP [Stergiou & Koubarakis, 00]

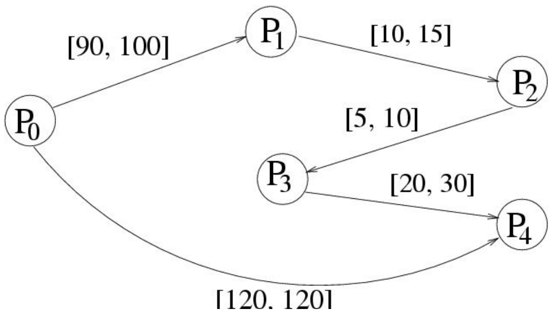
Solving the TCSP/DTP

- TCSP and DTP are NP-hard
- They are solved with backtrack search
- Every node in the search tree is an STP to be solved
- An exponential number of STPs to be solved ☹



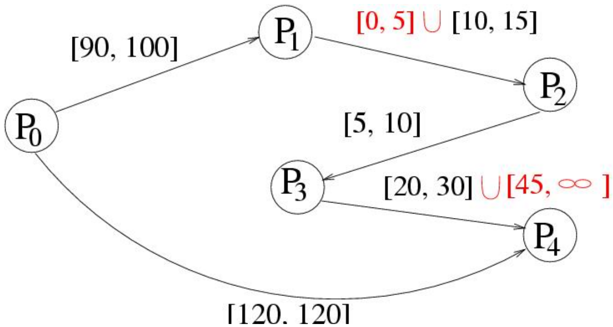


Exploit the structure..



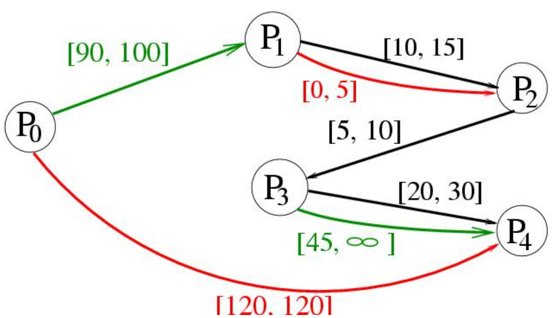
Simple Temporal Problem (STP)

- Floyd-Warshall, Bellman-Ford, DPC
- ΔSTP [Time 03]



Temporal CSP (TCSP)

- Search + ULT [Schwalb & Dechter 97]
- ΔAC, NewCyc, EdgeOrd [CP 03, AI Comm. 04]



Disjunctive Temporal Problem (DTP)

- Search + heuristics [S&K 00, O&C 00, Tsa&P 03]
- Some of our results are applicable

Algorithms for solving the STP

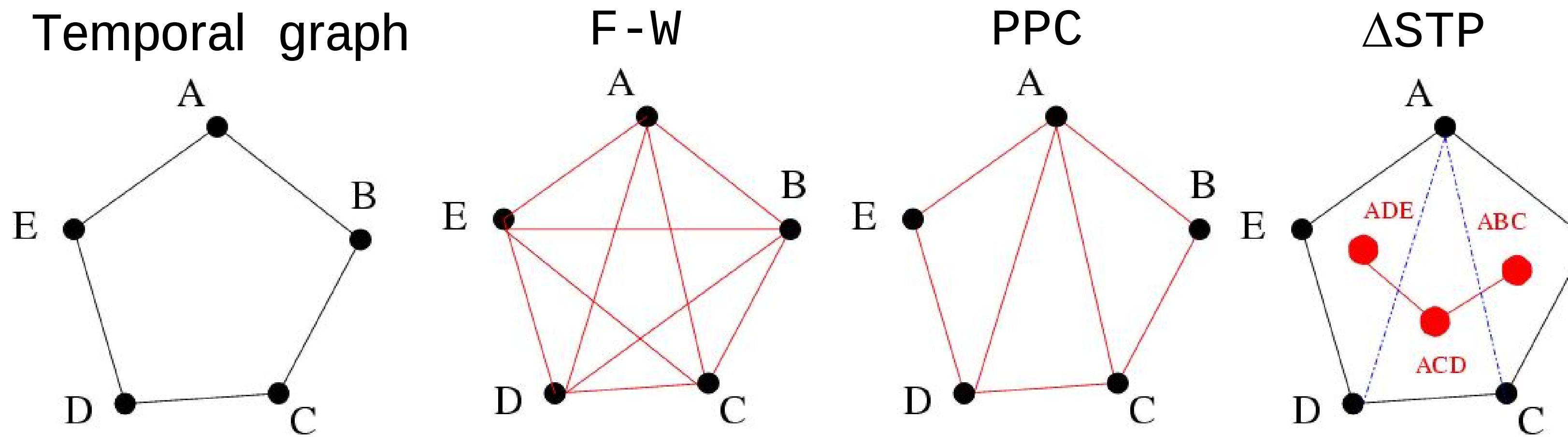
	Graph	Complexity	Consistency	Minimality
F-W	Complete	$\Theta(n^3)$	Yes	Yes
DPC [Dechter+, 91]	Triangulated	$O(nW^*(d)^2)$ very cheap	Yes	No
PPC [Bliek & S-H 99]	Triangulated	$O(n^3)$ Usually cheaper than F-W/PC	Yes	Yes
ΔSTP	Triangulated	Always cheaper than PPC	Yes	Yes
BF/incBF [Cesta & Oddi, 96]	Source point is added	$O(en)$	Yes	No

- **Consistency:** Determine whether a solution exists
- **Minimal network:** Make intervals as tight as possible

Partial Path Consistency (PPC)

- Known features of PPC [Bliek & Sam-Haroud, 99]
  - Applicable to general CSPs
  - Triangulates the constraint graph
  - In general, resulting network is not minimal
  - For convex constraints, guarantees minimality
- Adaptation of PPC to STP
  - Constraints in STP are bounded difference, thus convex, PPC results in the minimal network

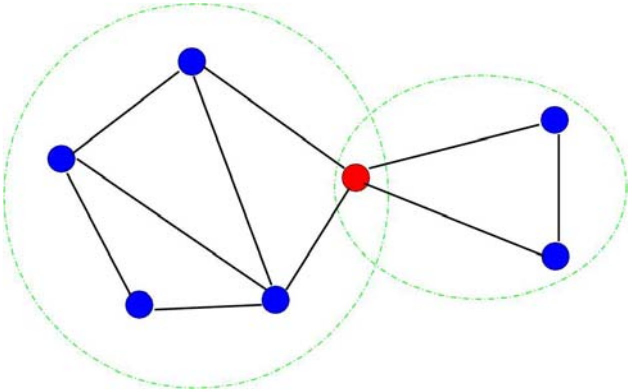
ΔSTP



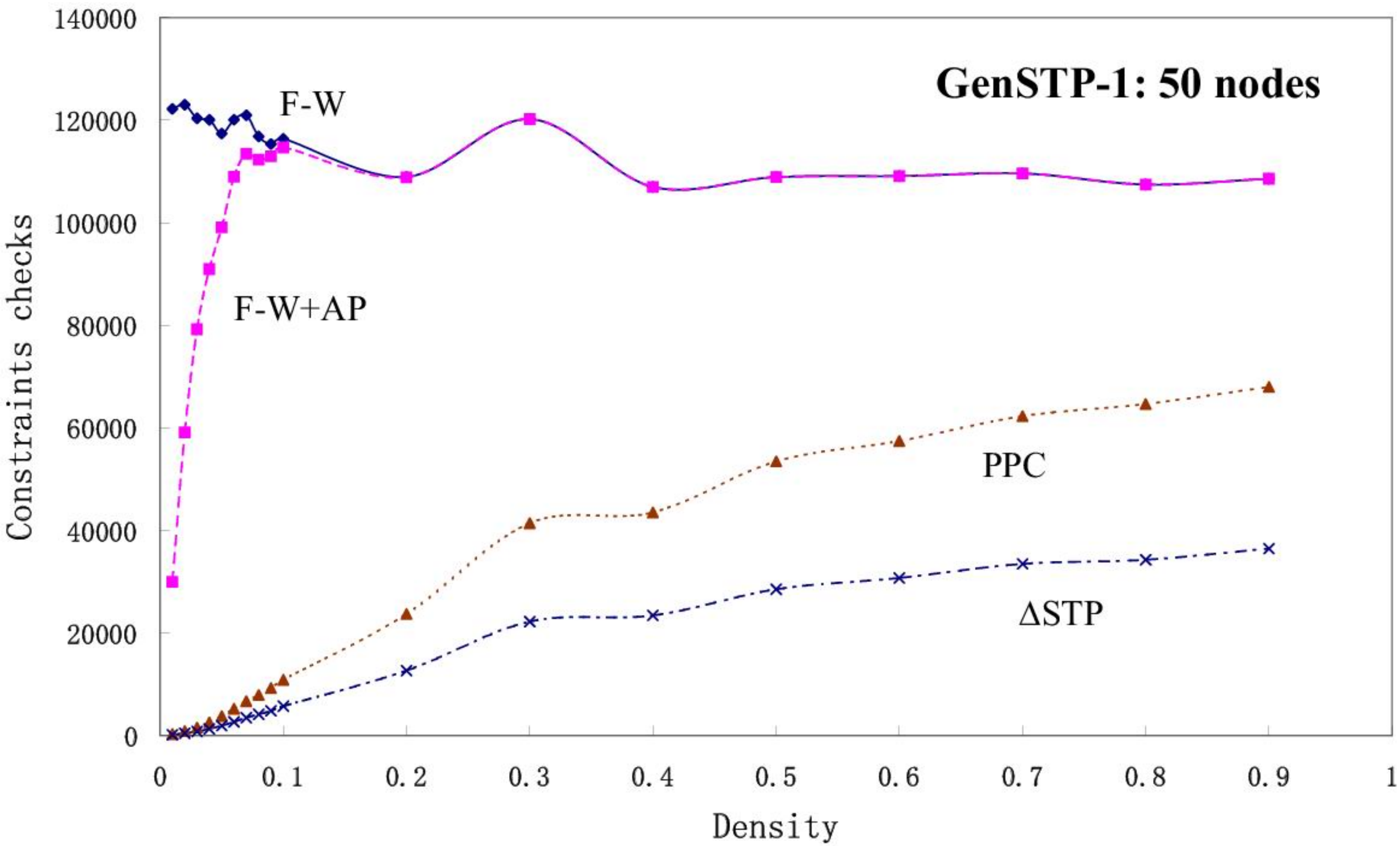
- ΔSTP is a refinement of PPC
  - Simultaneously update all edges in a triangle
  - Propagate updates through adjacent triangles

Advantages of ΔSTP

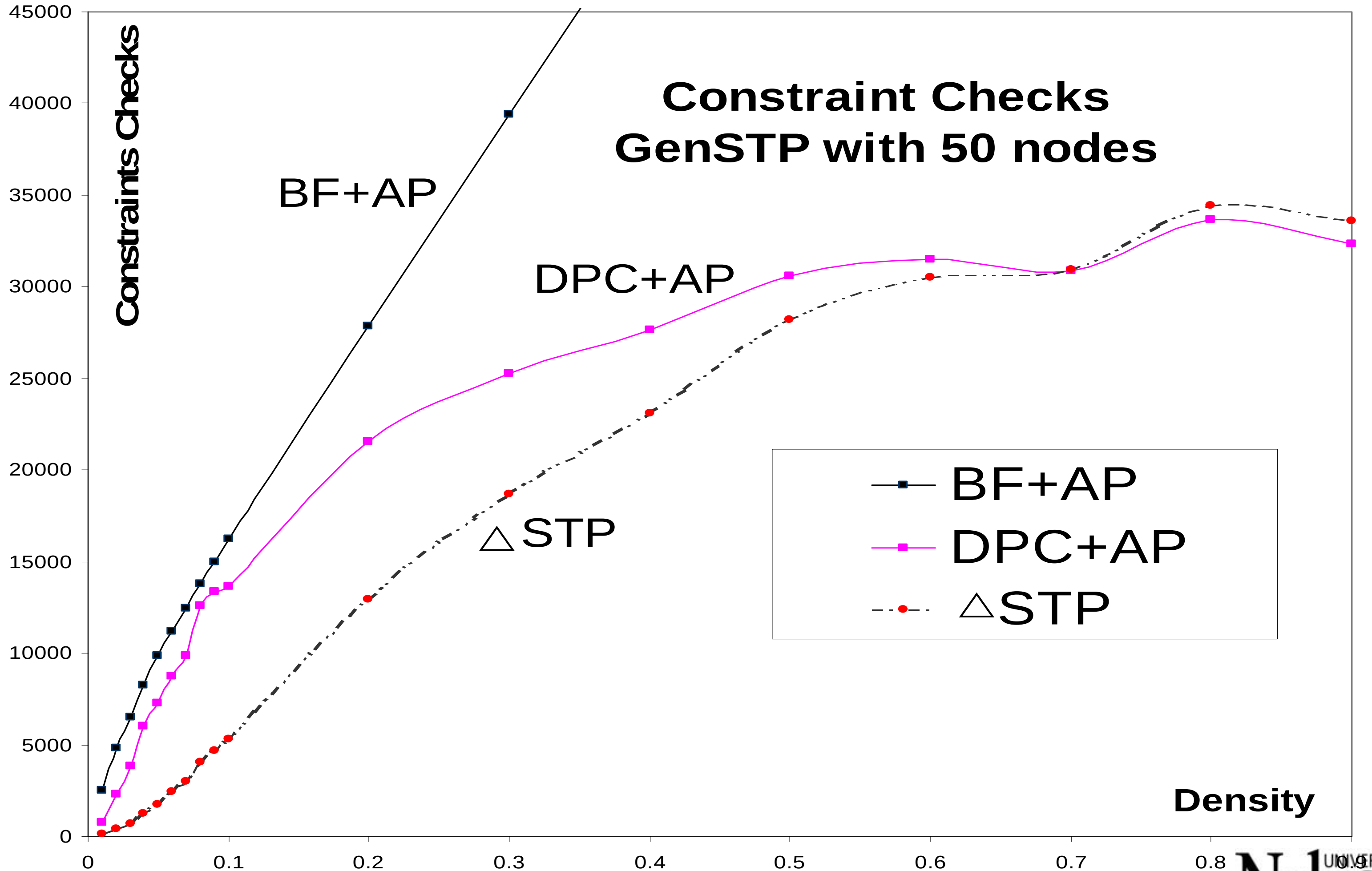
- Cheaper than `PPC` and `F-W`
- Guarantees the minimal network
- Automatically decomposes the graph into its bi-connected components
  - binds effort in size of largest component
  - allows parallellization
- Sweep through forth and back
  - Observed empirically, 2003
  - Explained by Nic Wilson @ 4C, 2005
  - Proved by Neil Yorke-Smith @ SRI, 2006



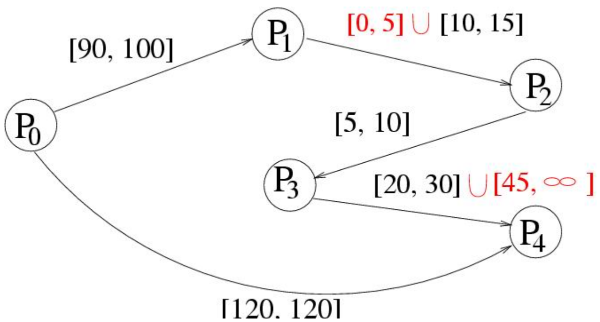
Finding the minimal STP

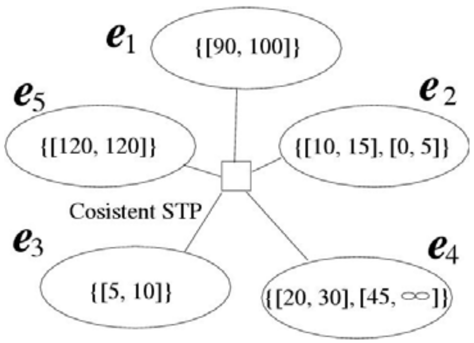


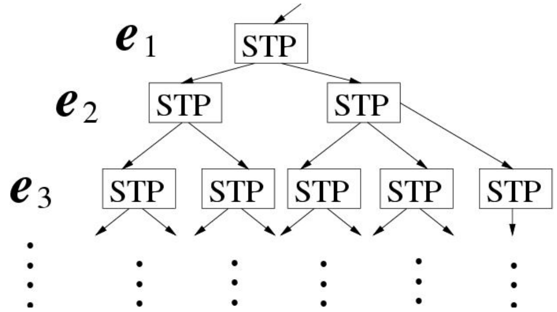
Determining consistency of the STP



Solving the TCSP [Dechter+, 00]



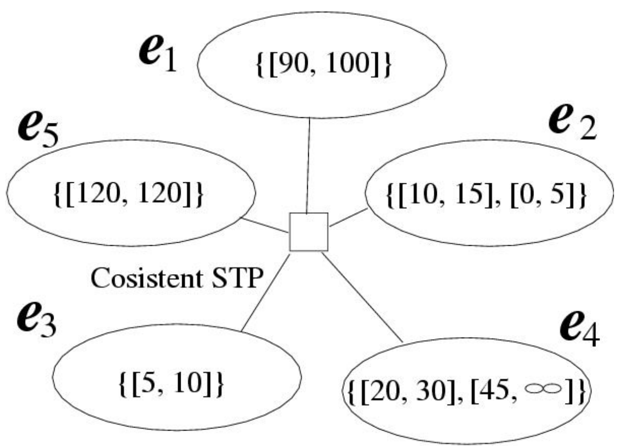


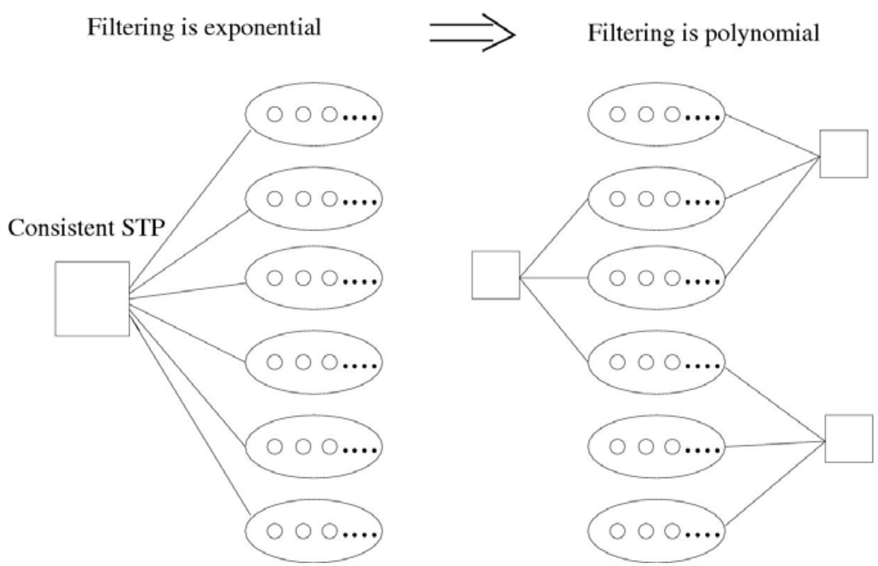


- Formulate TCSP as a meta-CSP
- Find all the solutions to the meta-CSP
- Use ΔSTP to solve the individual STPs efficiently
- But first, can we use some constraint propagation on the meta-CSP?

Preprocessing the TCSP

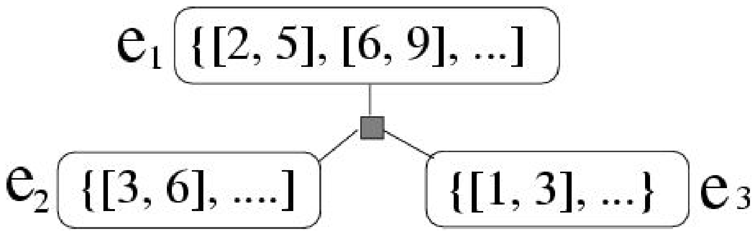
- Arc consistency
  - Single *n*-ary constraint
  - GAC is **NP**-hard
- ΔAC
  - Works on existing triangles
  - Poly # of poly constraints





ΔAC filters domains of TCSP

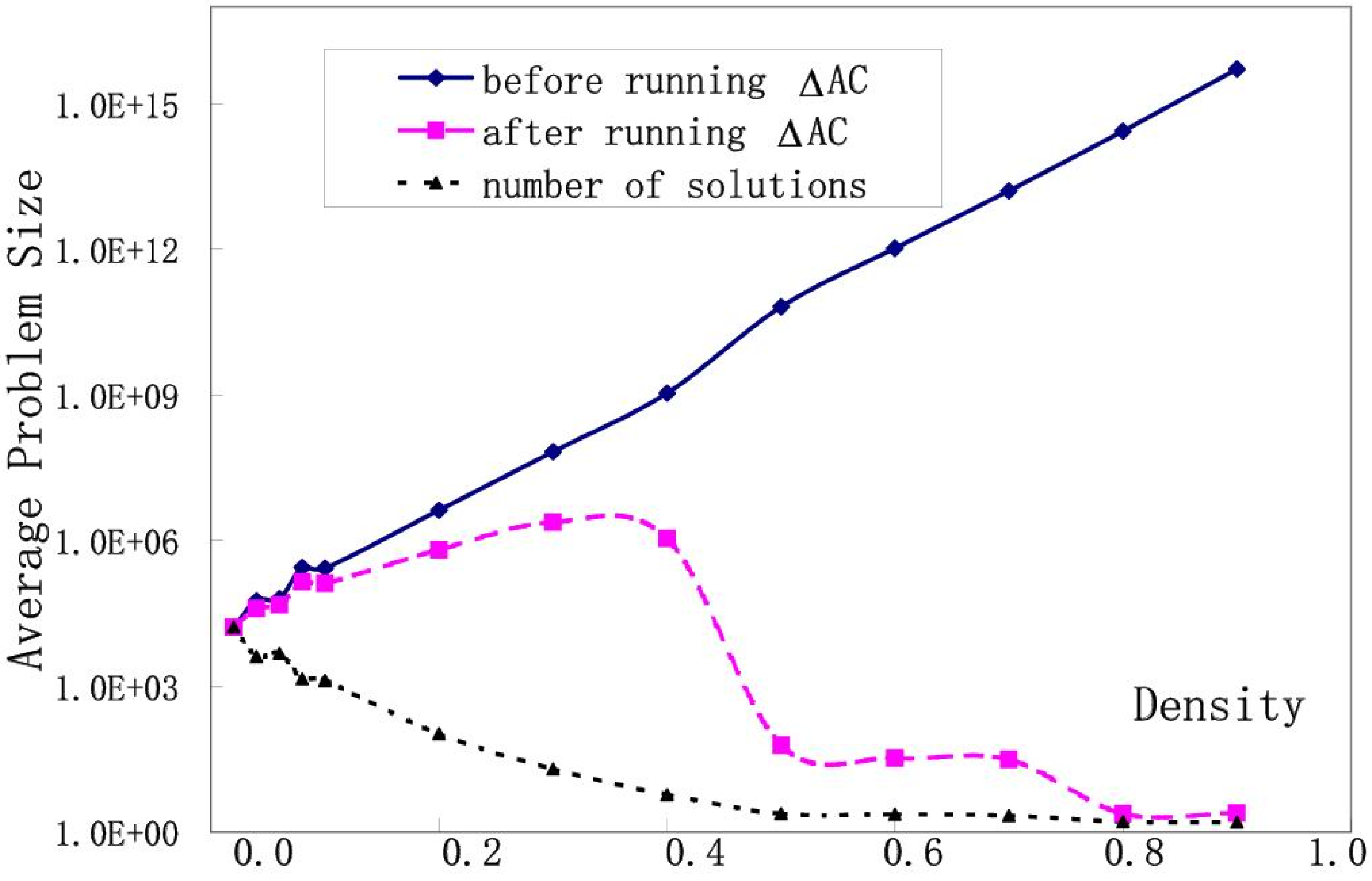
ΔAC removes values that are not supported by the ternary constraint



For every interval in the domain of an edge, there must exist intervals in the domains of the 2 other edges such that the 3 intervals verify the triangle inequality rule

- [1,3] in e_3 has no support in e_1 and e_2, ΔAC removes [1,3] from domain of e_3

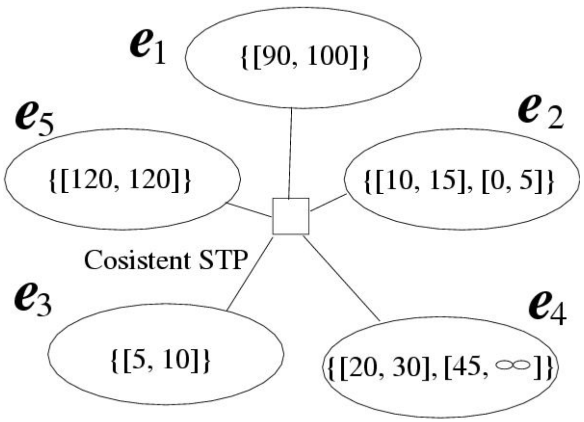
Reduction of meta-CSP's size

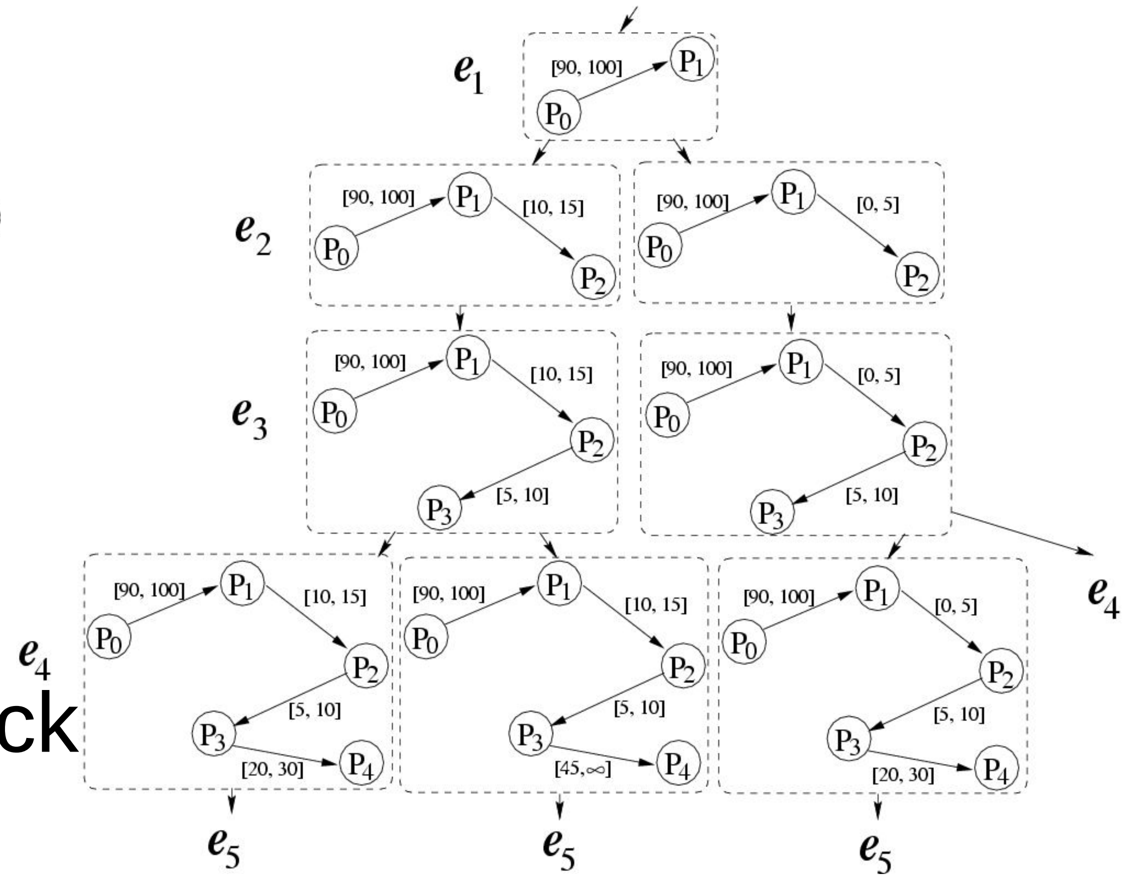


Advantages of ΔAC

- Powerful, especially for dense TCSPs
- Sound and cheap $O(n\ |E|\ k^3)$
- It may be optimal
  - Uses polynomial-size data-structures: `Supports, Supported-by` as in AC-4

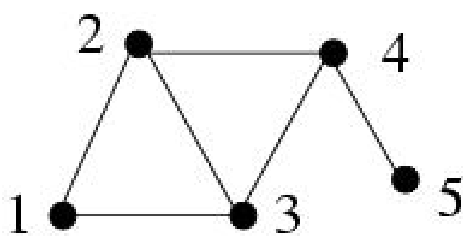
Improving search for the TCSP





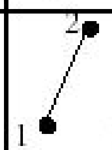
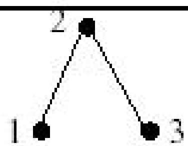
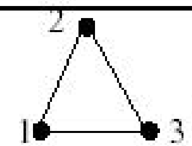
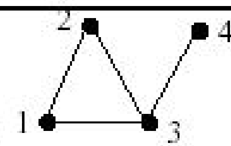
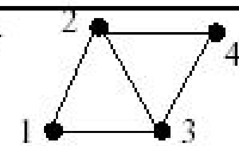
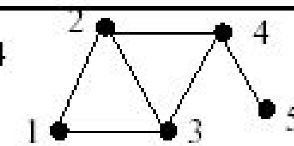
1. New cycle check
2. Edge Ordering

Checking new cycles: NewCyc



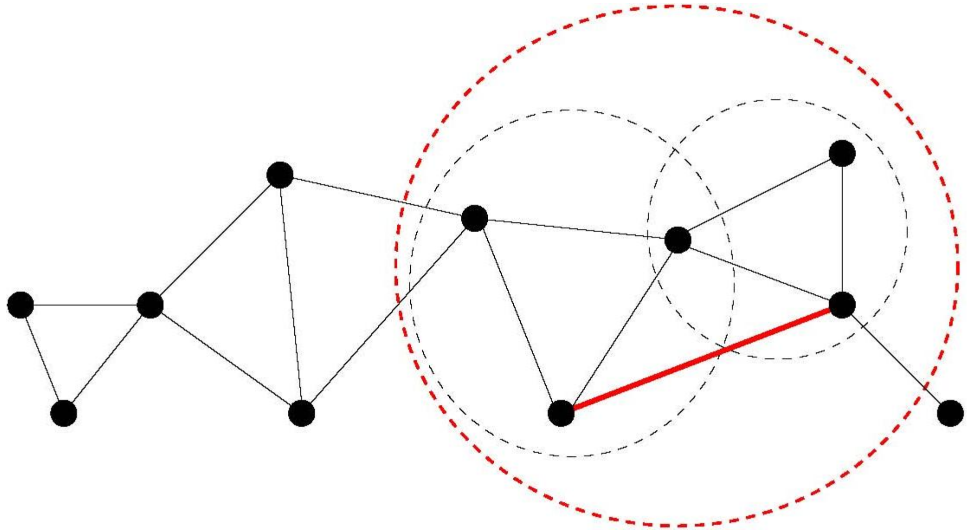
As a new edge is added at each step in search:

- Check the formation of new cycles $O(|E|)$
- Run ΔSTP only when a new cycle is formed

Search level	1	2	3	4	5	6	
STP							
Checking strategy							Total
Always	√	√	√	√	√	√	6
NC	×	×	√	×	√	×	2

Advantages of NewCyc

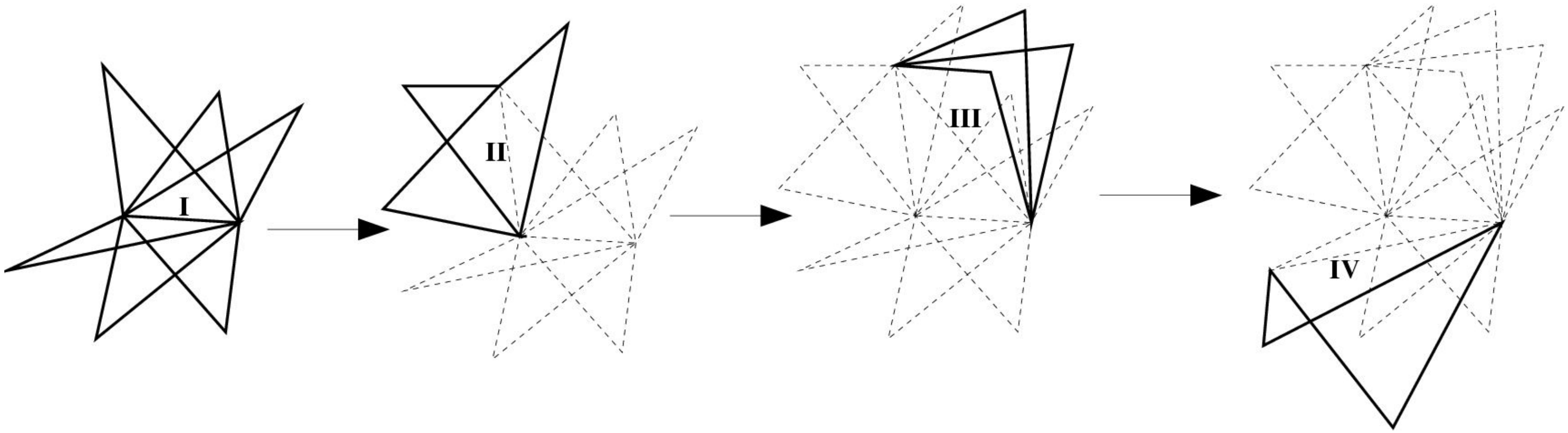
- Fewer calls to ΔSTP
- Operations restricted to **new** bi-connected component



- Does not affect # of nodes visited in search

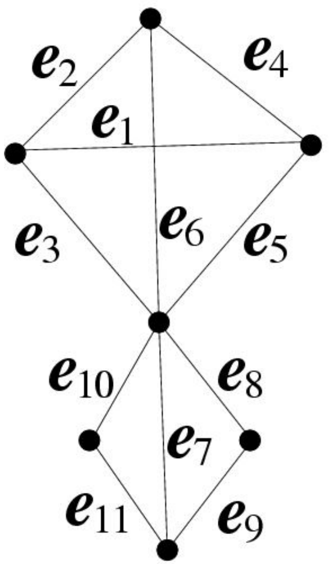
Edge ordering during search

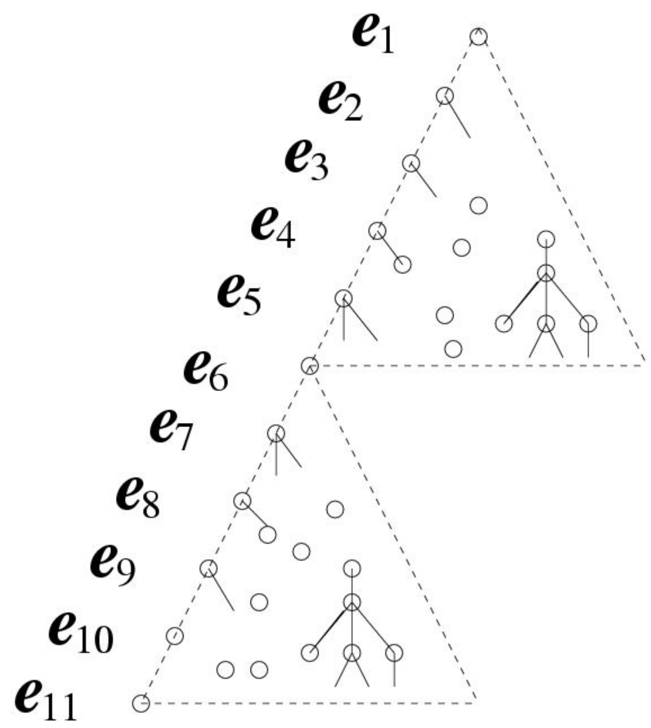
- Order edges using triangle adjacency
- Priority list is a by product of triangulation



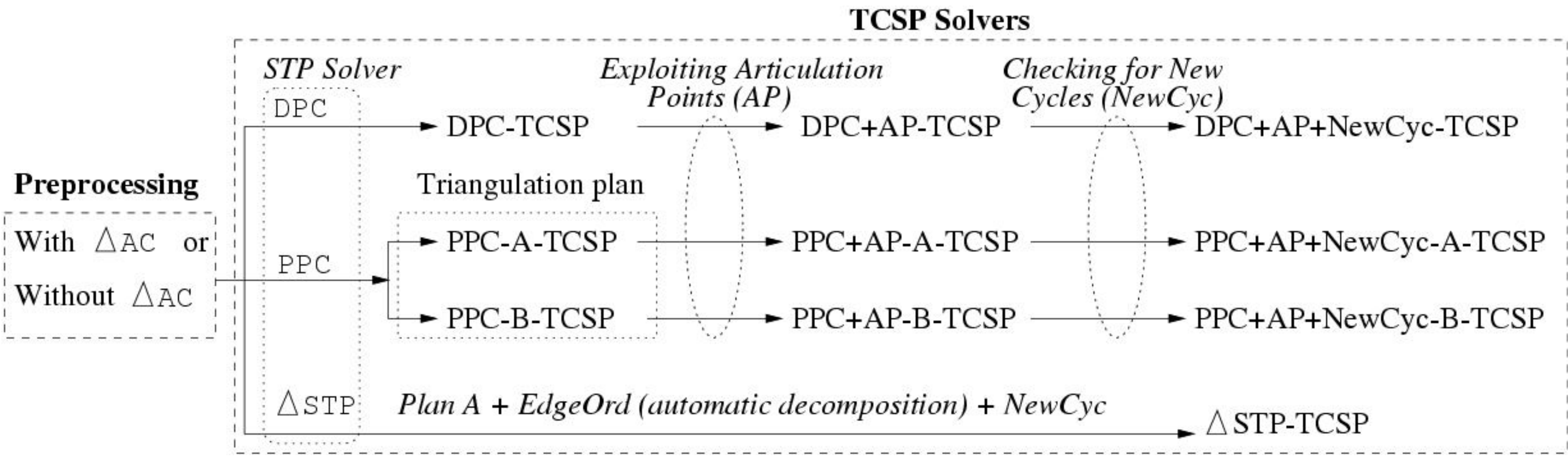
Advantages of EdgeOrd

- Localized backtracking
- Automatic decomposition of the constraint graph
  → no need for explicit detection of articulation points





Experimental evaluations



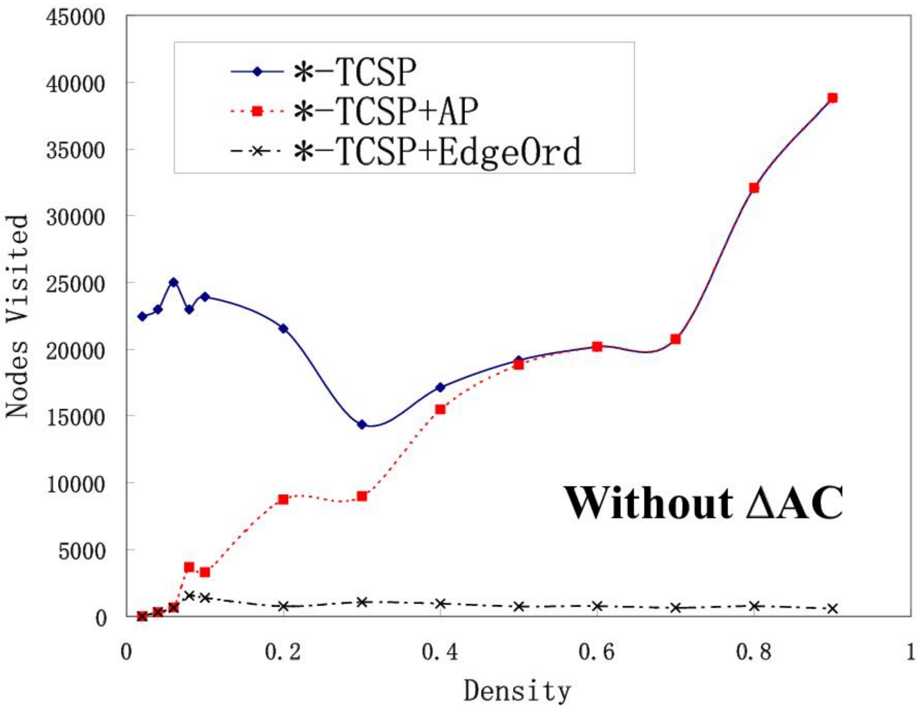
- New random generator for TCSPs
- Guarantees 80% existence of a solution
- Averages over 100 samples
- Networks are not triangulated

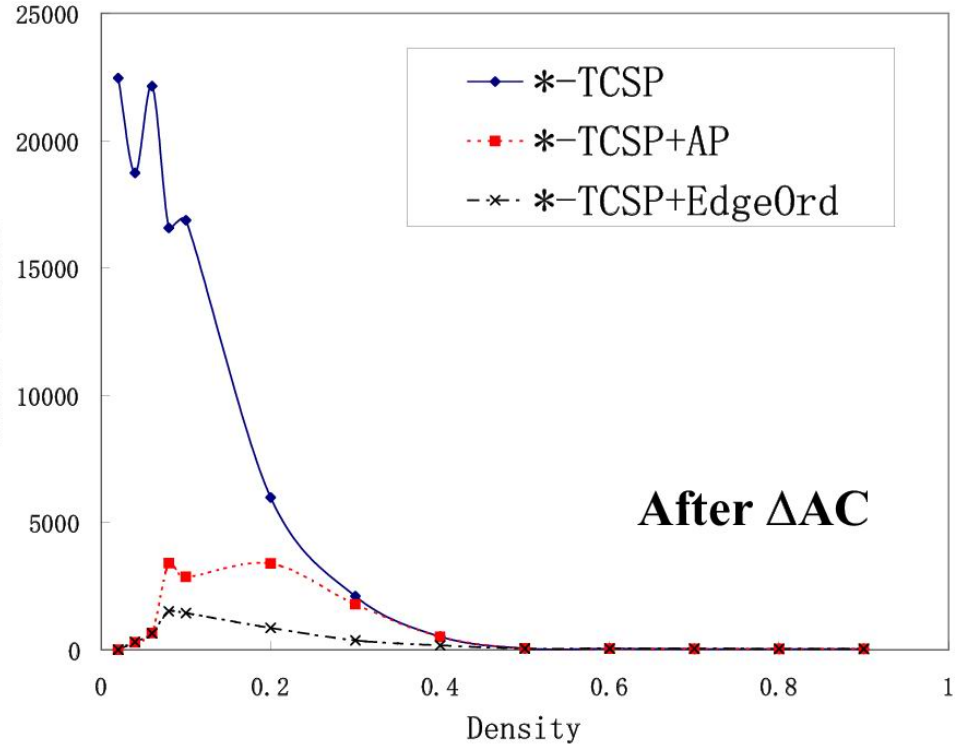
Expected (direct) effects

- Number of nodes visited (#NV)
  - ΔAC reduces the size of TCSP
  - EdgeOrd localizes BT

- Consistency checking effort (#CC)
  - AP, ΔSTP, NewCyc reduce number of consistency checking at each node

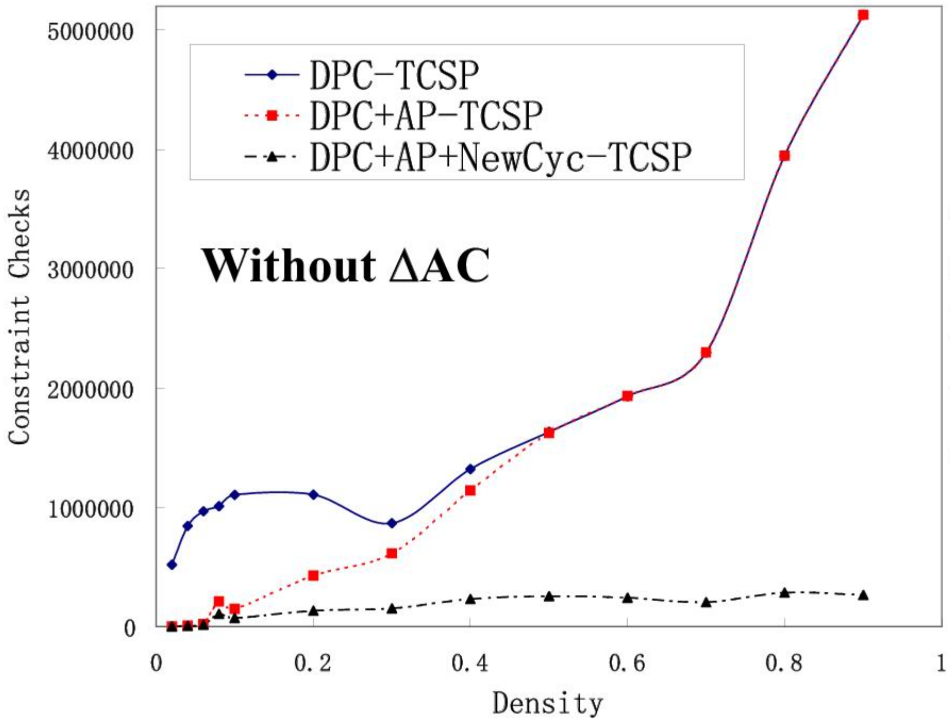
Effect of ΔAC on #nodes visited





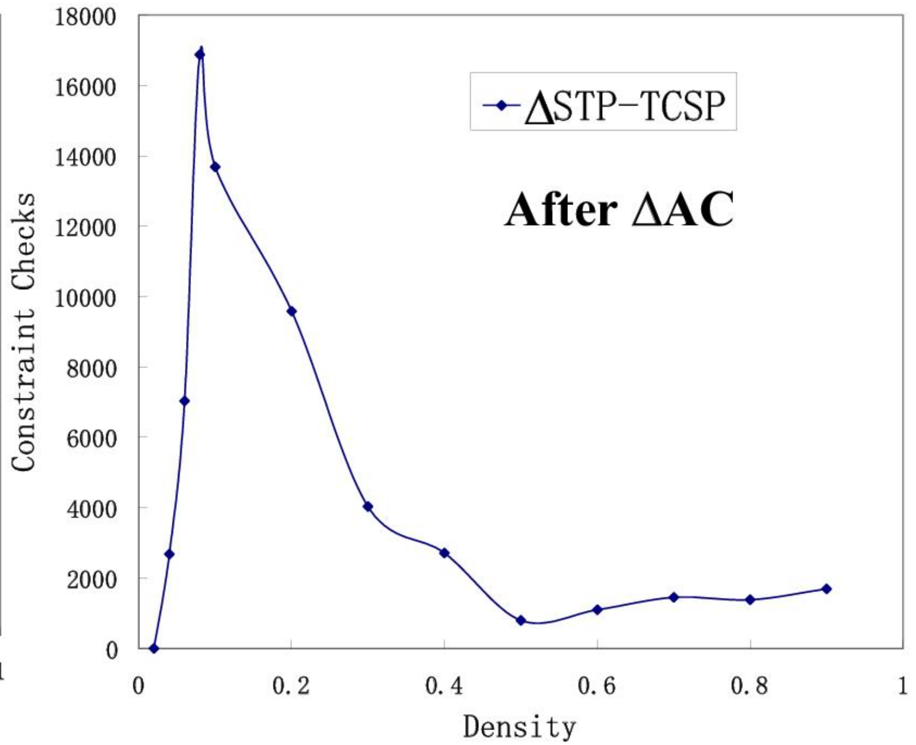
Cumulative improvement

Before, after AP, *after* NewCyc,…



Max on y-axis **5.000.000**

… *and now* (ΔAC, ΔSTP, NewCyc, EdgeOrd)



Max on y-axis **18.000, 2 orders of magnitude improvement**

Conclusions

- Contributions
  - ΔSTP, an (optimal?) algorithm for finding the minimal STP
  - ΔAC, an (optimal?) algorithm for filtering the TCSP
  - NewCyc, EdgeOrd heuristics for improving search on the TCSP
- Lesson… as usual
  - Exploiting the structure of the network and the properties of the constraints allow us to introduce new efficient techniques

Testing IncBF [Cesta & Oddi, 96]

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

Future work

- Use ΔAC as a look-ahead during search
- Investigate dynamic triangulation for dynamic edge ordering
- Investigate why IncBF of [Cesta and Oddi, 96] works so well with our heuristics NewCyc and EdgeOrd
- Test the results on the DTP

Pointers

- ΔSTP [TIME 03]
- ΔAC [AI Communications 04]
- Solving the TCSP [CP 03]

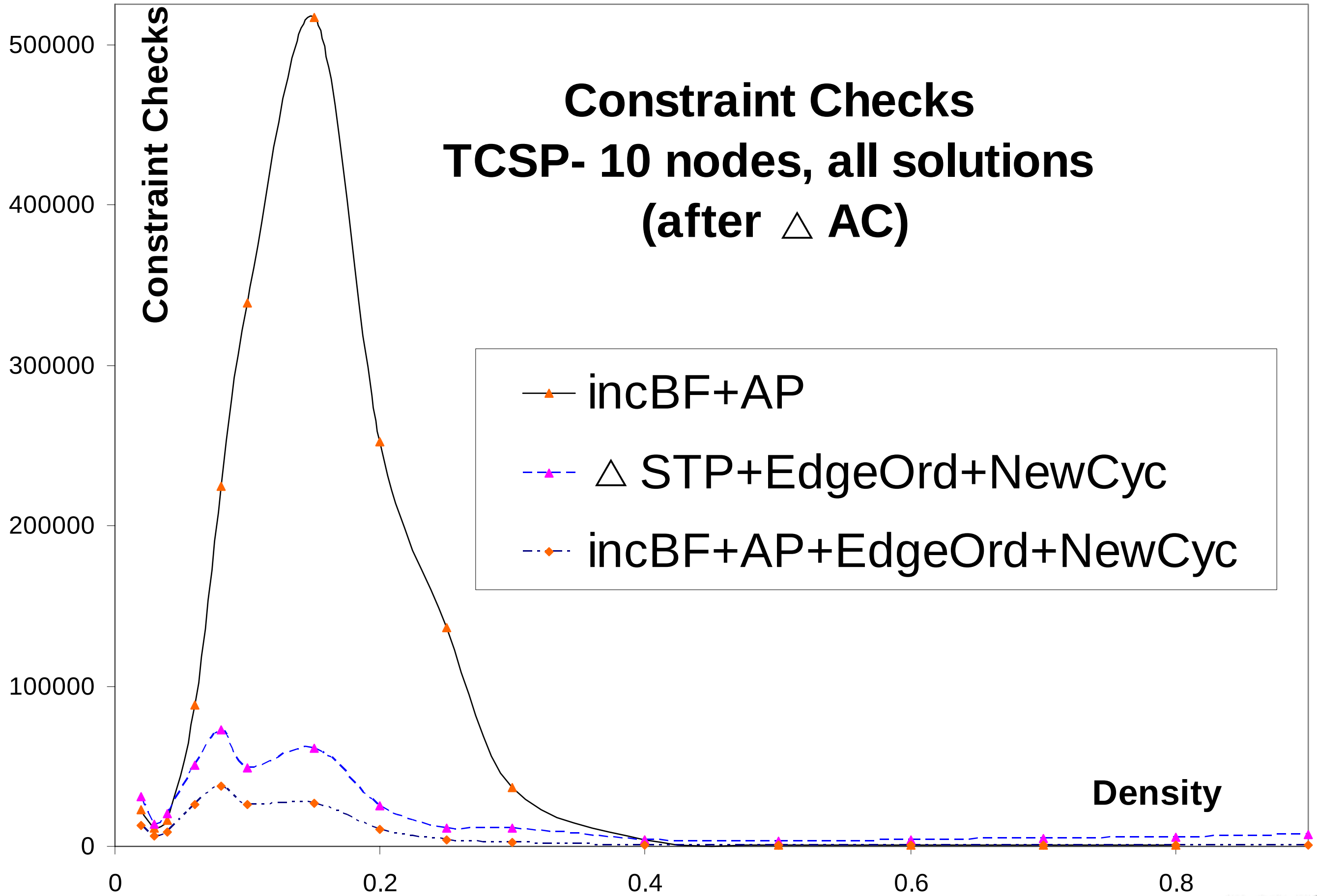
  (ΔSTP, ΔAC, NewCyc, EdgeOrd)
- ΔSTP vs. IncBF [Stdt ppr AAAI 04]
- All available from consystlab.unl.edu

Thank you for your attention

Time left for questions?

Additional slides

IncBF benefits from our work

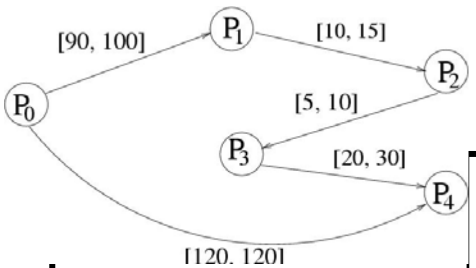


Computational problems

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

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

Algorithms for solving the STP



	Graph	Complexity	Consistency	Minimality
F-W	Complete	$\Theta(n^3)$	Yes	Yes
DPC [Dechter+, 91]	Triangulated	$O\ (nW^*(d)^2)$ very cheap	Yes	No
PPC [Bliek & S-H 99]	Triangulated	$O\ (n^3)$ Usually cheaper than F-W/PC	Yes	Yes
ΔSTP	Triangulated	Always cheaper than PPC	Yes	Yes
BF/incBF [Cesta & Oddi, 96]	Source point is added	$O\ (en)$	Yes	No

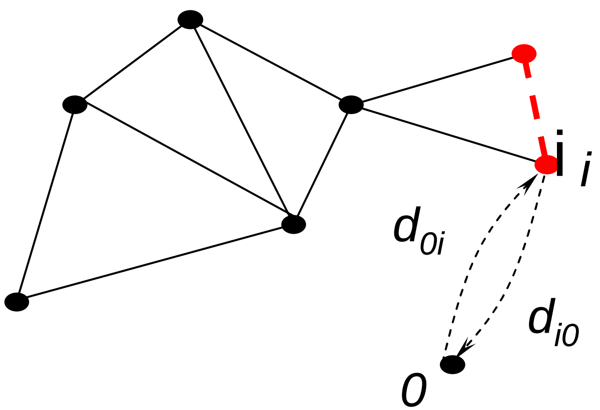
Summary of contributions

- ΔSTP, an algorithm for solving the STP
- ΔAC, a propagation algorithm for the TCSP
- Integration of the above with 2 new heuristics to improve search for solving the TCSP
  - New cycle check (NewCyc)
  - Edge ordering (EdgeOrd)
- Evaluations

An incremental version of BF (incBF)

When adding a constraint, incBF visits only nodes whose distance to origin is modified:

- Allows dynamic updates for both constraint posting & retraction.
- Localizes effects of change.
- Determines consistency of STP *by does not yield the minimal network.*
- Can detect inconsistency much earlier than BF by detecting negative cycles ($d0i + dio < 0$).
- 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