CONSONA: Constraint Networks for the Synthesis of Networked Applications

Lambert Meertens & Cordell Green

Asuman Suenbuel

asu@kestrel.edu

Stephen Fitzpatrick, Douglas Smith, Stephen Westfold

Kestrel Institute

Palo Alto, California

http://consona.kestrel.edu/

Q1: Technical Approach

- Develop *goal-oriented* techniques for modeling, designing and synthesizing NEST applications and service packages
  - express goals using time-based constraints
  - model solution methods/service packages as progress conditions on time-based constraints
- Co-design of applications & service packages
  - exploit context to optimize
    - multiple applications executing simultaneously over shared middleware
    - multiple service packages executing over shared communication
- Codify techniques as problem/solution taxonomies manipulated in automated composition and refinement tools
  - iteratively refine high-level goals into constraints satisfiable by known solution methods
- Generate optimized code for solution methods
  - use constraint propagation & maintenance techniques to optimize communication & to direct searches

Example: UAV swarm

Step 1: State the problem

- Assumptions:
  - UAVs communicate through wireless broadcasts
  - range is limited (scalability!)
  - signal strength can be used to estimate distance
- Safety requirements:
  - *vehicles must maintain safe distances*
- Progress requirements:
  - *observe given area*
  - *collect information in a timely manner*
- “Non-functional” requirements:
  - *minimize energy expenditure*

Refining requirements: Maintain safe distance

Step 2: refine problem statement by strengthening constraints

- System-wide constraint:
  - safe distances => projected flight cones should not intersect
- This constraint can be maintained by adjusting the flight paths
  - => maintain knowledge of relative positions, velocities, …

Step 3: refine system-wide constraints into local form

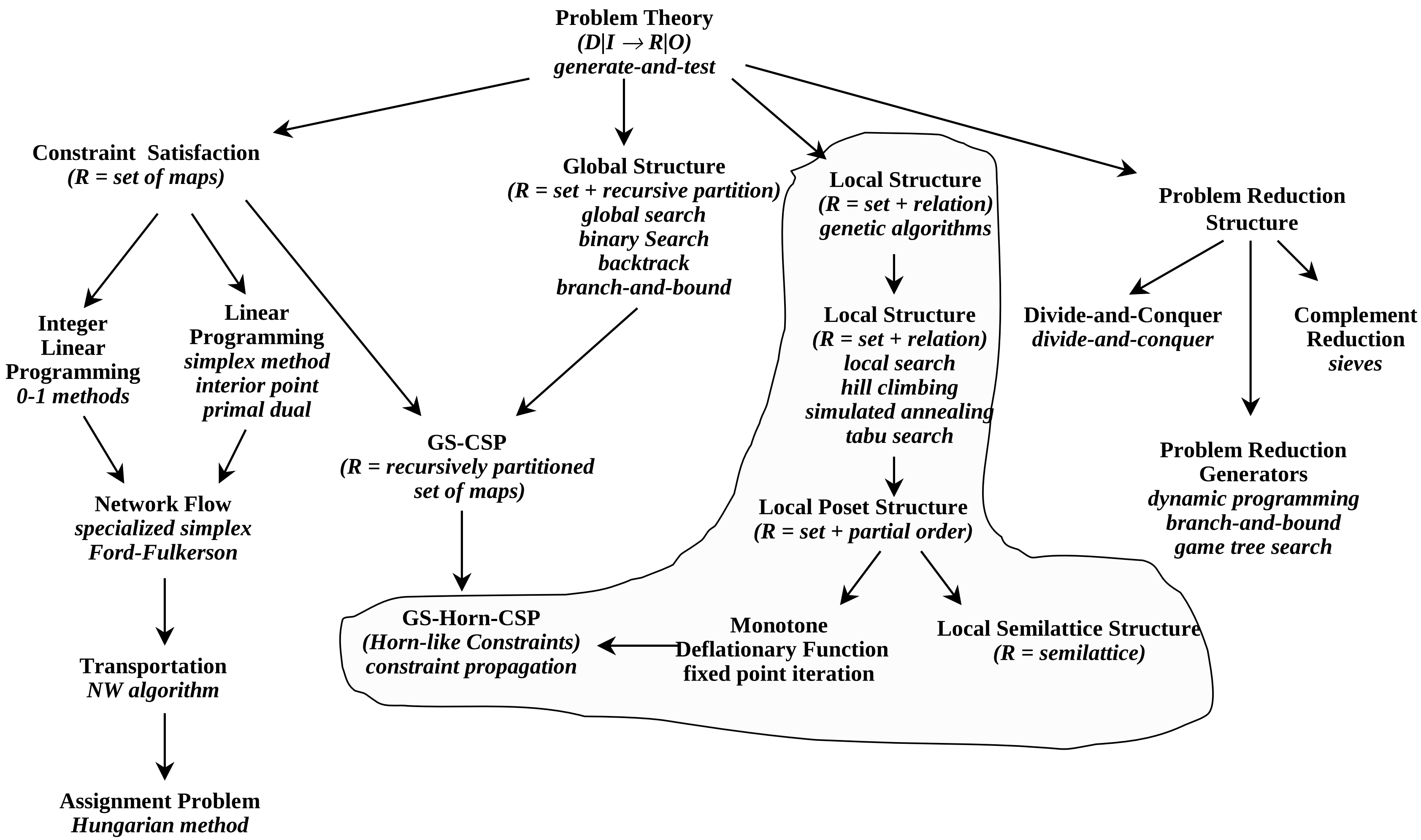
- System-wide constraint => *distributed* constraint *network*:
  - each UAV has a map of some other UAVs’ positions
  - each UAV’s map must be consistent with observed signal strengths
- Constraint network can be maintained by each UAV adjusting estimated positions
  - need to maintain inter-map consistency as local adjustments are independently made
    - this is an instance of the general requirement of consistency in distributed knowledge!

Generating Code

Step 4: optimize communication & searches

- Maintenance of constraint network => local variable updates
  - local variable updates must be propagated
    - optimization restricts propagation to needed information & to needed recipients
  - local variable updates must be coordinated
    - stochastic, local algorithms
    - self-stabilizing algorithms

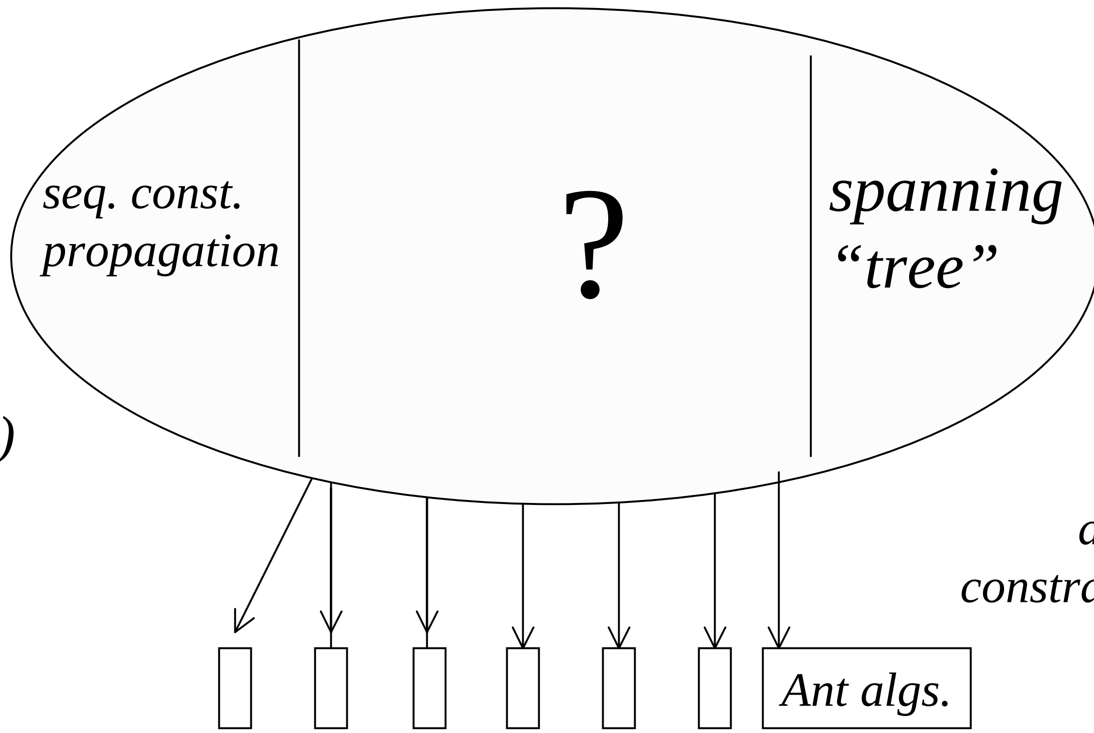
Inspiration: Taxonomy of Algorithm Theories



Extension to Distributed Constraints

Problem → global constraints + local platform capabilities

sequential algorithms (traditional algorithms design)



self-stabilization

protocol transformers

distributed local-repair

distributed constraint propagation

….variants + combinations of algorithms…

Q2: Product type

- Middleware
- Application software
  - Berkeley OEP
- Algorithms/theoretical foundations
  - Methods for developing self-stabilizing algorithms and design patterns for distributed systems
- Tools
  - Integrated modeler-generator

Q3: NEST Technology Areas

- Coordination services
  - models, specifications of NTP etc.
- time-bounded synthesis
  - distributed anytime algorithms for constraint satisfaction
- service composition and adaptation
  - composition of service packages and applications
  - context dependent optimization

Q4: Challenge Area Classification

a) **Primary: Lifecycle -** our research creates design time tools and methods for generating efficient runtime code based upon self-stabilizing algorithms

b) **Secondary: Solution domain -** our technology supports co-design of applications and middleware

c) **Solution domain issues**: our technology addresses

- Primary:
  - online reconfiguration
  - probabilistic methods
- Secondarily:
  - offline configuration (pre compilation)
  - (fault tolerance)
- Could be applied to:
  - time synchronization
  - group membership & consensus

Q5: Initial Collaboration Plan

a) OEP collaboration
   - Berkeley OEP

b) Group 1 collaboration:
   - technical exchange with XEROX PARC
     - constraint algorithms
     - component-based expertise exchange
   - open for other groups
     - formal modeling of existing middleware e.g NTP
     - developing stochastic local algorithms for distributed constraint satisfaction

Q6: Integration Interface and Opportunities

a) Provide:
   - framework for constraint based specification of service packages
   - tools for composition and refinement
   - Generic patterns and taxonomy for distributed self-stabilizing algorithms

b) Need:
   - standard Berkeley APIs
   - initially NTP and point-to-point communication

Q7: OEP Framework Requirements

- On board clock
- Development environment
  - compiler, debugger
  - profiling tools, simulator
- Sensors & effectors
  (we are considering an application involving distributed beam focusing)
  - photo-sensors
  - actuactors for mirrors

Q8: Scalability

- Number of nodes:
  - ~10^5
- Node memory
  - application dependent
- Other specific scalability issues
  - scalable communication mechanism
    - e.g. local multicast rather than global broadcast
  - scalable application

Q9: Training Requirements

- What knowledge is needed by researchers trying to integrate with/ use your group technology?
  - some understanding of first-order logic and temporal logic
  - expressing application specifications as directed constraints