

# Improving Coalition Performance by Exploiting Phase Transition Behavior

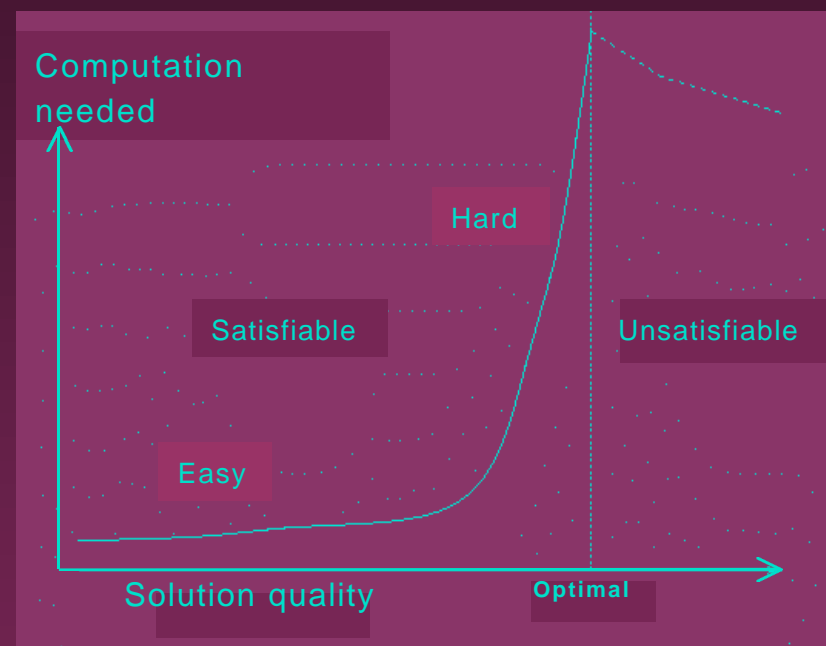
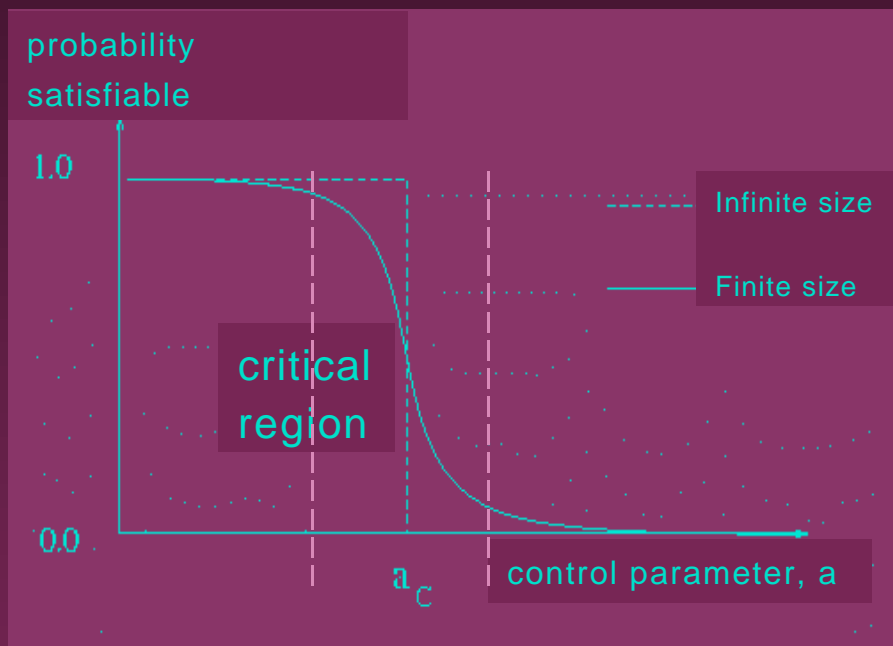
**David W. Etherington  
Matthew L. Ginsberg  
Andrew J. Parkes**

**Computational Intelligence  
Research Laboratory  
University of Oregon**

# Outline

- Background: Phase transitions and optimization
- Summary of CIRL approach
- Relation of CIRL approach to CAMERA/SNAP
  - ◆ coalitions (and orderings) of sorties
- Results
  - ◆ preliminary model with SNAP-like search performance
  - ◆ phase transitions and incomplete knowledge problems
  - ◆ reasoning maintenance and arithmetic constraints
- Future plans

# Phase Transitions



- Problem character changes in the *critical region*
- Computational cliff: Improvements become expensive

# Coalition Control

Coalitions will affect search performance.

**Goal:** Guide coalition formation/maintenance

- broad measures of coalitions formation/change
- hill-climbing methods to drive better coalition sizes/boundaries

**Applicability:**

- sufficient variability for coalition choices to be important
- not all variability/structure can be picked up statically

**Benefits:** Improved robustness by

- recovery from bad initial coalition formations
- appropriate (re)allocation of computational resources

# Decomposition Advisor Agent

Provide locally useful information

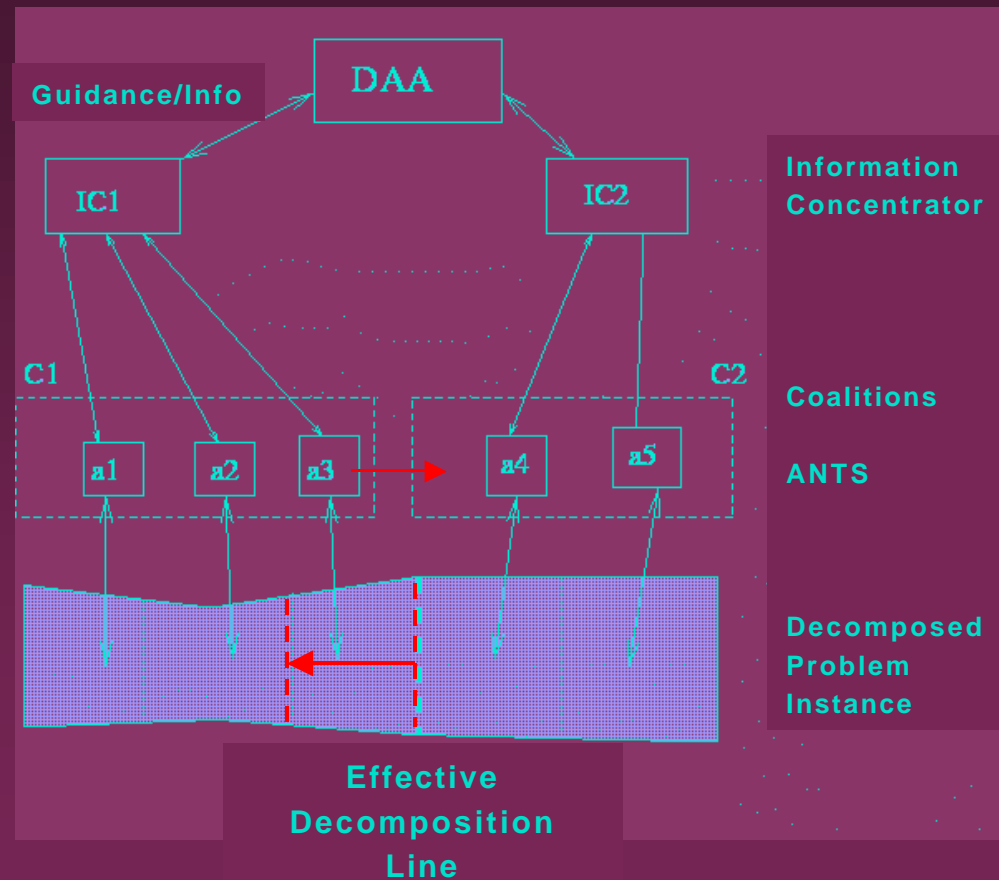
- compatible with global phase transition properties

Get info from concentrators

- suggest incremental changes to improve decomposition
- e.g., move  $a_3$  to  $C_2$

Info comes from a larger view than individual ANTs should have

- impractical to retrofit it to individual ANTs



# Guidance Information

Use information potentially available from ANTs.

1. Counting-based measures of imminent cliffs:
  - count constraints
  - measure bottleneck resource usage
2. Collection of implied constraints (“no-goods”)
  - count no-goods – use to predict imminent cliffs
  - structure of no-goods – use to adjust coalitions

# CIRL Approach: Summary

Understand interactions between

- search method
- phase transitions
- coalition boundaries
- emergence and structure of no-goods

Use to

- guide coalition formation and refinement
- guide search and computational resource allocation
- predict search performance

# Collecting No-Goods

## Dependency/Reasoning Maintenance

- detect imminent computational cliffs
- dynamic discovery of effective constraint graph
  - ◆ detect mismatch between coalitions and graph

## Methods to be studied:

- extend ANT system(s) to generate/store no-goods
- use “negotiation histories”
  - ◆ “who did you talk to?”
  - ◆ “who gave you the most trouble?”

Encapsulate into “Anytime No-good Detector (ANDE)”

# CIRL Approach & SNAP

SNAP: Extensive work on domain constraints

- mission selection
  - ◆ hard constraints: e.g. forced missions, prerequisites
  - ◆ otherwise optimize for combat readiness, etc
- mission formation
  - ◆ mostly hard constraints, some optimization (risk,...)

Search using “One sample search” (1-samp):

- negotiation-based construction of a single solution using greedy domain-dependent heuristics

# Coalitions and SNAP

Sorties currently constructed and published in unstructured groups.

Potential:

- carefully guided coalitions of sorties
- other opportunities for coalitions of
  - ◆ pilots
  - ◆ planes
  - ◆ ranges

# Sampling-Based Search Issues

Require quantitative measures for

- performance prediction: relevant to “soon enough”
- performance improvement: relevant to “good enough”

Improving the search:

- improve 1-samp itself
- use multiple sampling methods (many good methods are known)

Cases:

- Intra-Sample: when does solution improvement become hard?
- Inter-Sample: what improvement will more samples give?

# Results

1. Artificial domain and “intra-sample” SNAP-like search
2. Phase transitions and incomplete knowledge problems
3. Dependency “no-good” methods exploiting arithmetic

# A Simplified Domain

Simplified artificial domain to reduce cognitive/coding overheads

- grid of variables – c.f. resource vs. time
- localized constraints – gives more realistic structure
- switches,  $s(i)$ , to activate sets of constraints, c.f. missions/sorties

if  $s(23)$  then (local constraints on grid)

Maximize number of switches ON

- such that active constraints are all satisfiable
- c.f. maximize number of sorties that can be scheduled

# Search Method

“If sortie can be scheduled then commit, else reject.”

Set all switches to OFF.

Select an ordering for the switches.

For each  $s$  in ordered list of switches:

    if  $s=ON$  is allowed (active constraints satisfiable)

    then

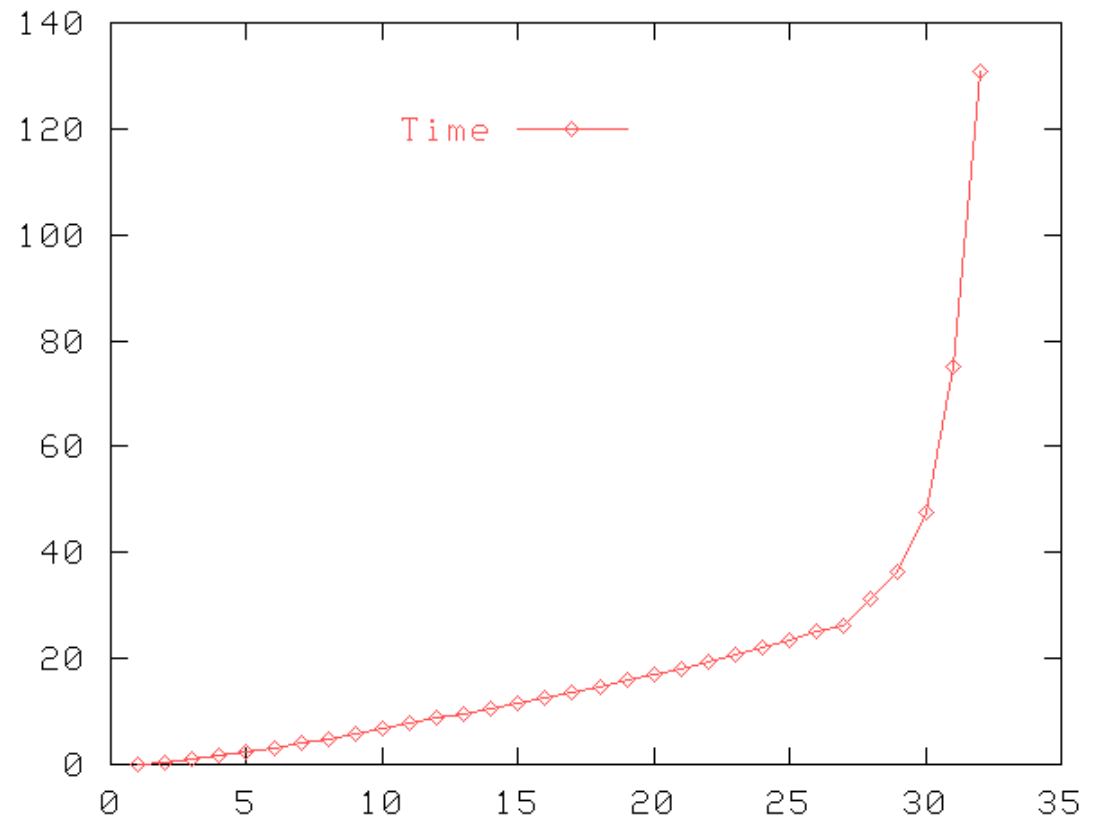
        commit  $s := ON$

# SNAP-like Probe: “Easy-Hard-Stuck”

Phase transition (PT) corresponds to 26-29 switches.

PT predicts switch failures start at 26-29.

Final “stuck” point is in deep unsat region of PT.



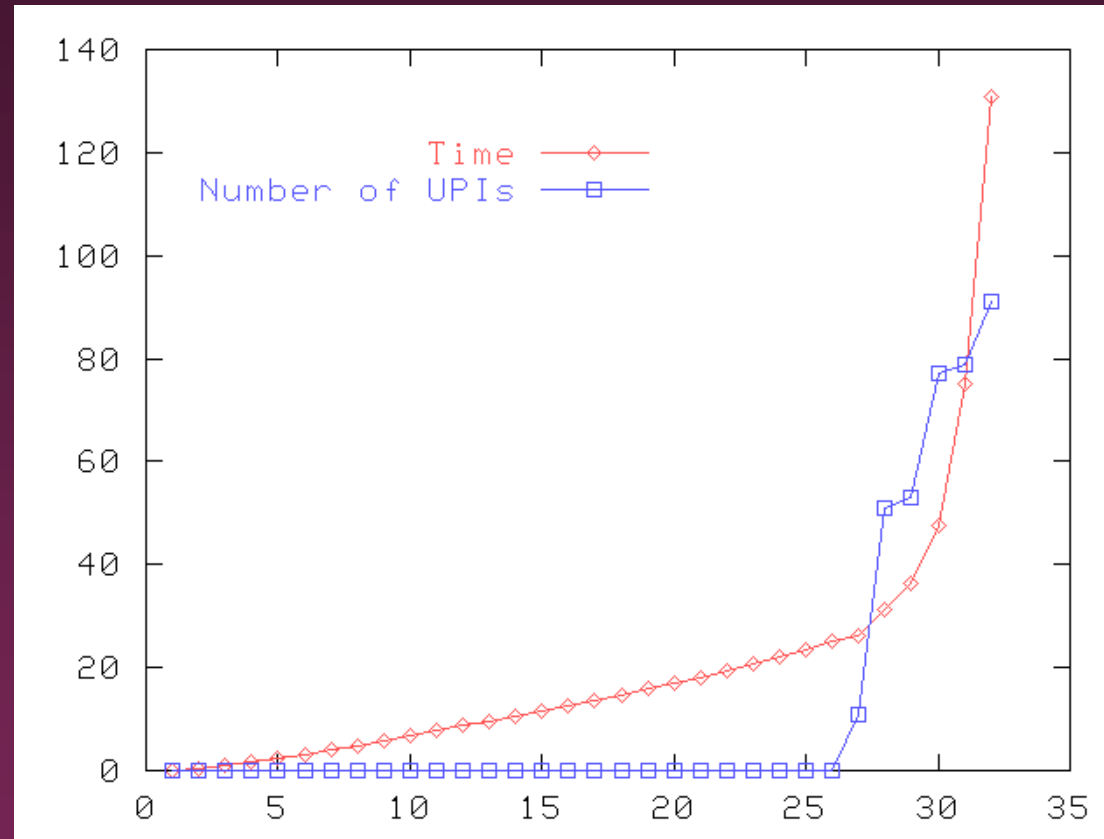
Number sorties/switches turned ON

# Unary-Prime-Implicates (UPIs)

UPIs are variables with single legal value remaining.

- UPIs emerge near easy-hard transition
- Increase rapidly during hard phase

Supports usage for cliff predictions.



Number sorties/switches turned ON

# Dependency-Maintenance

Dependency/reasoning maintenance methods collect no-goods.

E.g. Relevance-Bounded Learning for Satisfiability (RELSAT)

- successful for SAT problems
- keeps only relevant no-goods
- polynomial space use
- natural decoupling of independent subproblems

Techniques being adapted to

- optimization problems
- representations suitable for resource bounded problems

# SAT/CSP vs. Arithmetic

Should use arithmetic reasoning when doing resource problems:

- SAT/CSP representations known to be bad at counting
- math programming representations better
- Pseudo-Boolean (PB) representation:
  - ◆ linear inequalities on 0/1 variables

PARIS: Pseudo-Boolean RELSAT

- extract no-goods to encapsulate reasons for search failure
- compose no-goods from different failed branches of search
  - ◆ developing methods to select most useful compositions

Relevance:

- direct usage of PARIS as ANT for subproblems
- lessons in composing (arithmetical) reasons for search failure

# Issues in Incomplete Knowledge

ANTs generally do not have complete knowledge

Phase transition work has assumed complete knowledge.

For incomplete knowledge problems

- can phase transitions occur?
- are they important?

# Minesweeper

Mines placed randomly on grid

If select clear square

Then obtain number of neighboring mines

Else boom.

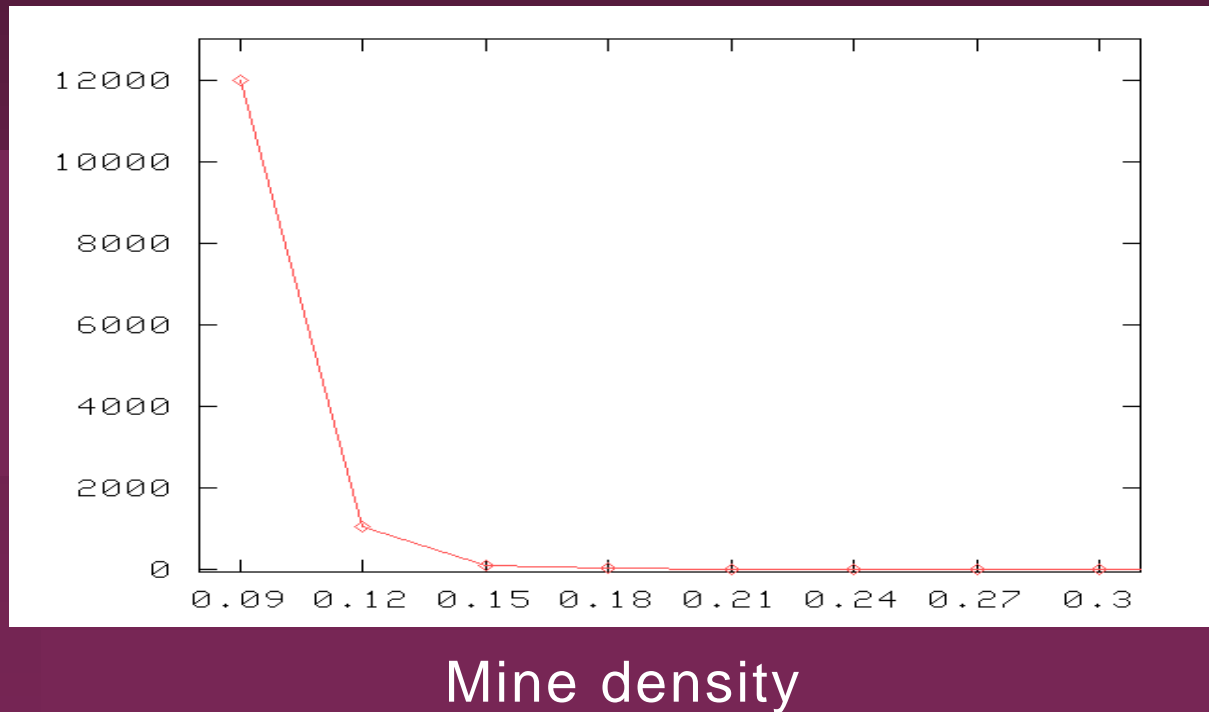
Incomplete knowledge but can do propagation:

2	1		Mine
1	1	1	
0	0		Clear

# Thresholds in Propagation Distance

Minesweeper: Long thin grid, give info on one narrow edge and measure distance it propagates.

Distance  
cleared using  
simple  
propagation



# Are such transitions important?

## Individual ANTs

- do not have complete access to properties of rest of system
- might use statistical estimates of such external properties

## Phase transition

- utility of different reasoning levels can have sharp dependence on statistical properties

## Goal: predict best levels of reasoning

- avoid wasting effort with over-powerful reasoning methods
- avoid ineffectiveness of under-powered reasoning methods

# Plans

- Extend simple artificial domain
  - ◆ capture more of the SNAP search behavior
- Use domain (and SNAP itself) to
  - ◆ understand what determines shape of performance curve
  - ◆ develop **quantitative** predictions of search performance
  - ◆ explore wide range of coalition and search control methods
- Extract and analyze “negotiation histories” to dynamically
  - ◆ discover effective interactions (constraint graphs)
  - ◆ refine coalitions and the next/current search sample

Pick best of the methods to encode in DAA guidance agent

Put in hooks in SNAP to interface with CIRL DAA

Extend lessons/DAA to other ANT systems

# Summary

Use phase transitions to provide theoretical and experimental bases to evaluate coalition structure

- too large, too small, misaligned?

Developing ways to

- Detect imminent computational cliffs
- Predict performance curves

Derive mechanisms for building/using good coalitions

- develop methods that improve decompositions