Communication and Computation in DCSP Algorithms

Universitat de Lleida

Cèsar Fernàndez
Ramón Béjar

Intelligent Information Systems Institute
CORNELL

Bhaskar Krishnamachari
Carla Gomes

September, 10th 2002

Testing DisCSP algorithms in real environments

The need for realistic DisCSP benchmarks

- We need problem instances that capture the structure of real DisCSP
- But we need an easy way to generate instances with different constrainedness

Our proposal: a distributed resource allocation problem

The effect of the communication network

- In real scenarios, agents will experience the delays of the communication network
- We should investigate how delays affect the performance of the algorithms

Our proposal: a discrete event-simulator able to simulate different delay distributions

Contents

- DisCSP
- SensorDCSP benchmark
- DisCSP algorithms
- Complexity profiles of DisCSP algorithms on SensorDCSP
- The effect of the communication network load
  - Delays actively added by the Agents
  - Delays because of the network load

Distributed Constraint Satisfaction Problem (DisCSP)

Agents need to solve different CSPs

- Local constraints (within Agents)
- Global constraints (between Agents)

How do Agents solve it ?

- Local constraints: computation
- Global constraints: computation + **communication**

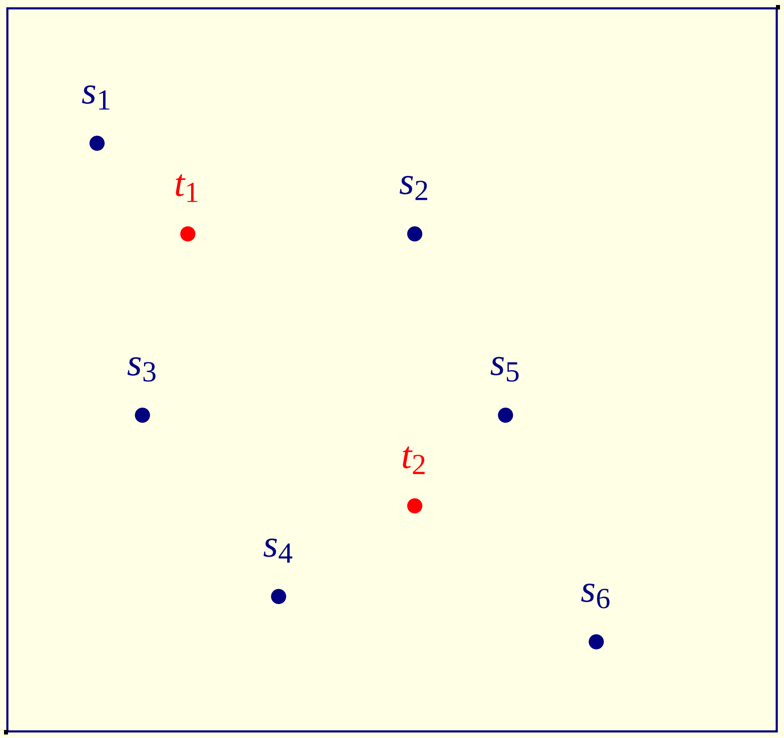
What is the impact of the **communication network** in the performance ?

SensorDCSP benchmark

Input:

- Multiple sensors (s_i) and mobiles (t_j) randomly scattered
- Constraints:
  - sensor visibility bipartite graph (P_v): sensors that can track a given mobile
  - every sensor can track at most one mobile
  - sensor compatibility graph (P_c): compatibility relation between sensors

Objective: Assign to each mobile 3 pair-wise compatible sensors that can track it

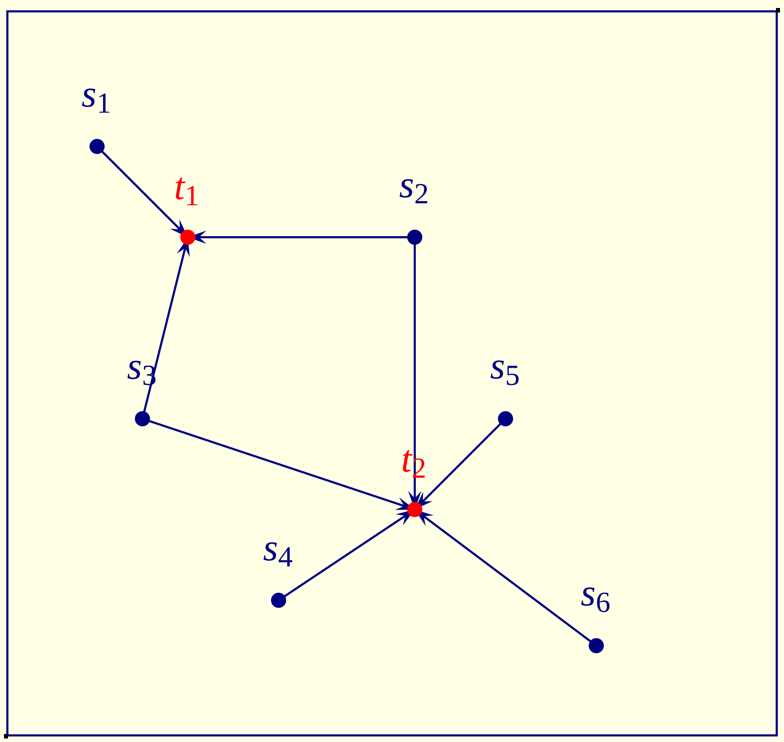


SensorDCSP benchmark

Input:

- Multiple sensors (s_i) and mobiles (t_j) randomly scattered
- Constraints:
  - sensor visibility bipartite graph (P_v): sensors that can track a given mobile
  - every sensor can track at most one mobile
  - sensor compatibility graph (P_c): compatibility relation between sensors

Objective: Assign to each mobile 3 pair-wise compatible sensors that can track it

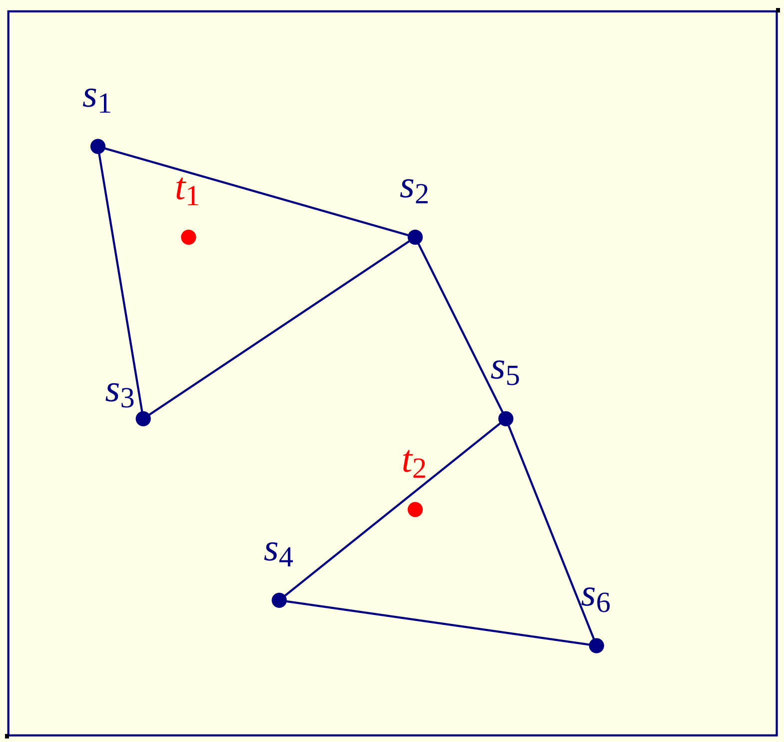


SensorDCSP benchmark

Input:

- Multiple sensors (s_i) and mobiles (t_j) randomly scattered
- Constraints:
  - sensor visibility bipartite graph (P_v): sensors that can track a given mobile
  - every sensor can track at most one mobile
  - sensor compatibility graph (P_c): compatibility relation between sensors

Objective: Assign to each mobile 3 pair-wise compatible sensors that can track it

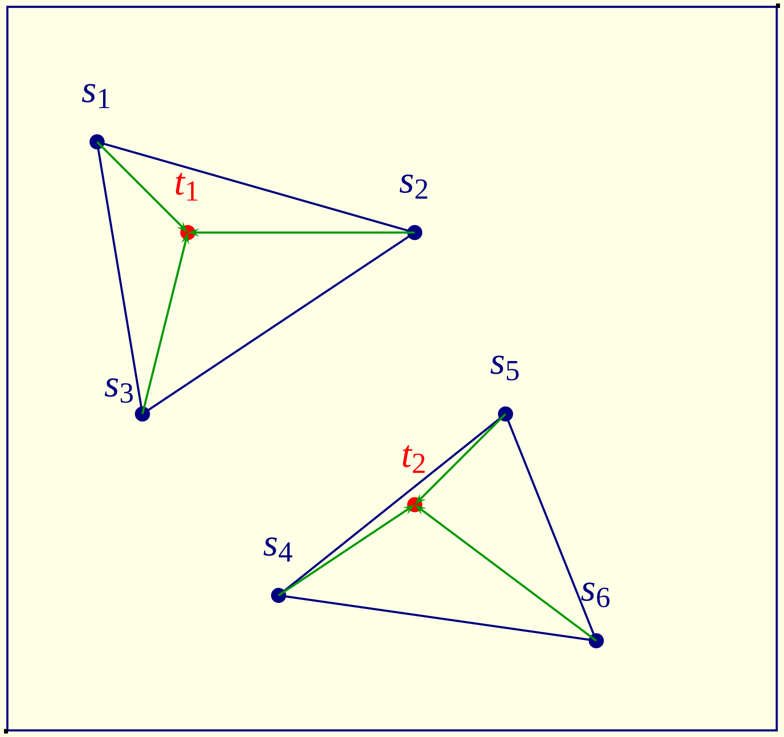


SensorDCSP benchmark

Input:

- Multiple sensors (s_i) and mobiles (t_j) randomly scattered
- Constraints:
  - sensor visibility bipartite graph (P_v): sensors that can track a given mobile
  - every sensor can track at most one mobile
  - sensor compatibility graph (P_c): compatibility relation between sensors

Objective: Assign to each mobile 3 pair-wise compatible sensors that can track it



DisCSP algorithms

Messages exchanged:

- **Ok**: inform about assignments to its neighbors
- **nogood**: ask a neighbor to change its assignment

Algorithms:

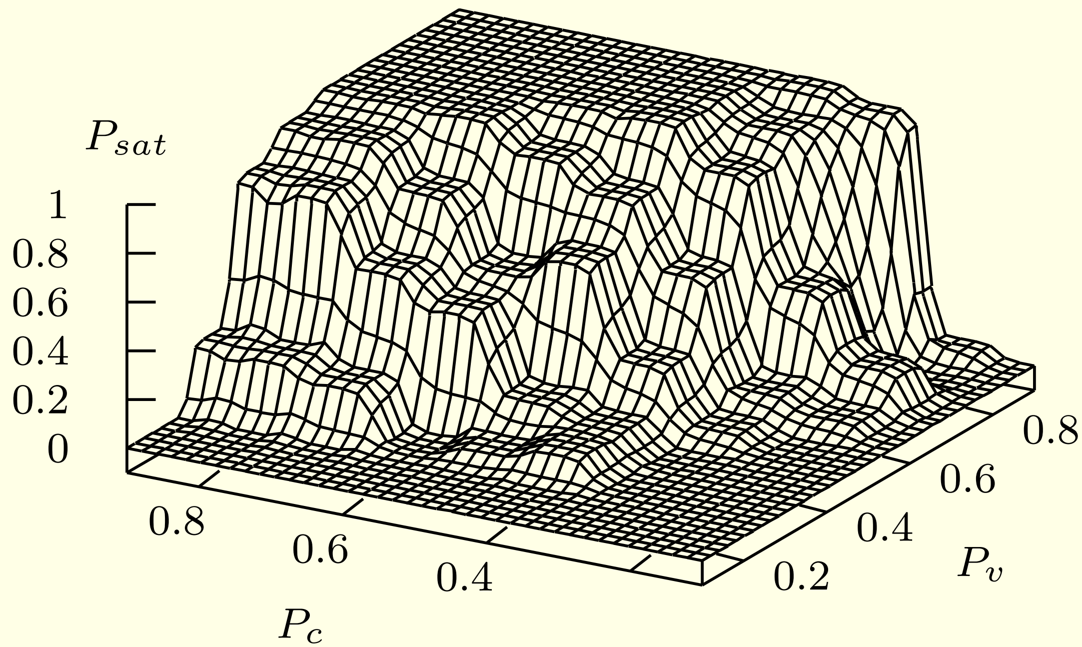
- Asynchronous Backtracking (ABT)
  - static ordering between agents
  - random selection among values consistent with higher (priority) agents
- Asynchronous Weak-Commitment Search (AWC)
  - dynamic priority ordering
  - random selection among values consistent with higher agents and that minimize constraint violations with lower agents

Implemented using a **distributed discrete-event network simulator**

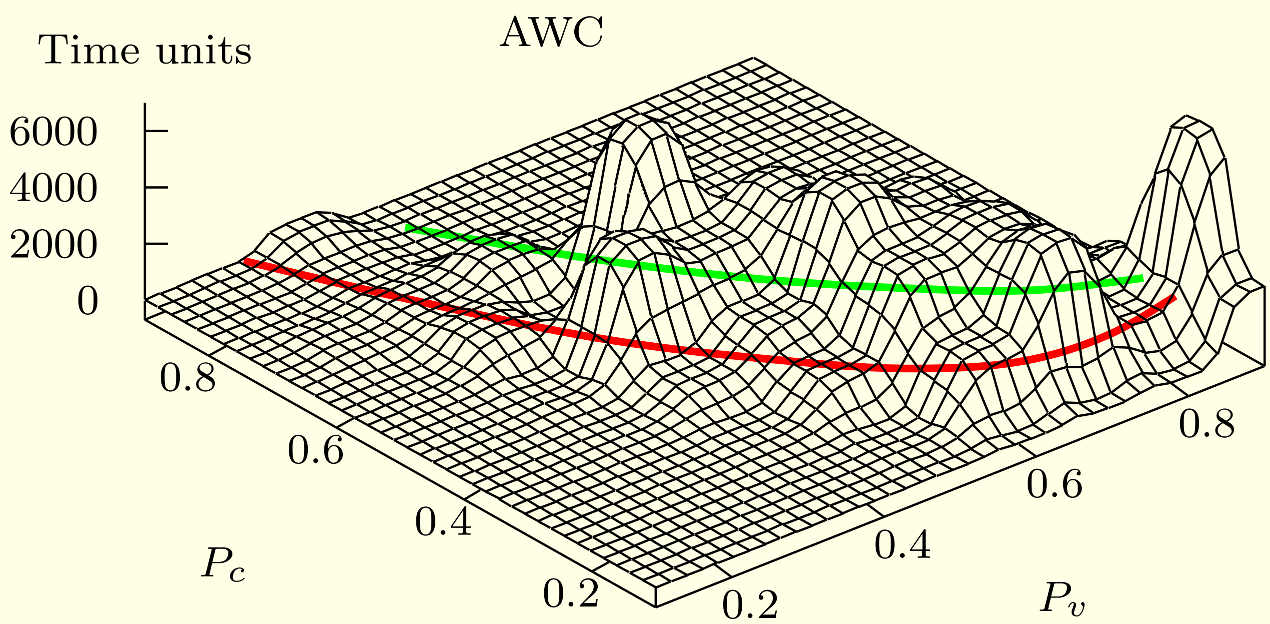
Complexity profiles of DisCSP algorithms on SensorDCSP

- Experimental scenario
  - Inter-agent communication delays: exponentially distributed, mean 1 time unit
  - 3 mobiles and 15 sensors. P_c and P_v ranging from 0.1 to 0.9
  - 19 instances per point. 9 independent runs for instance

Ratio of satisfiable instances

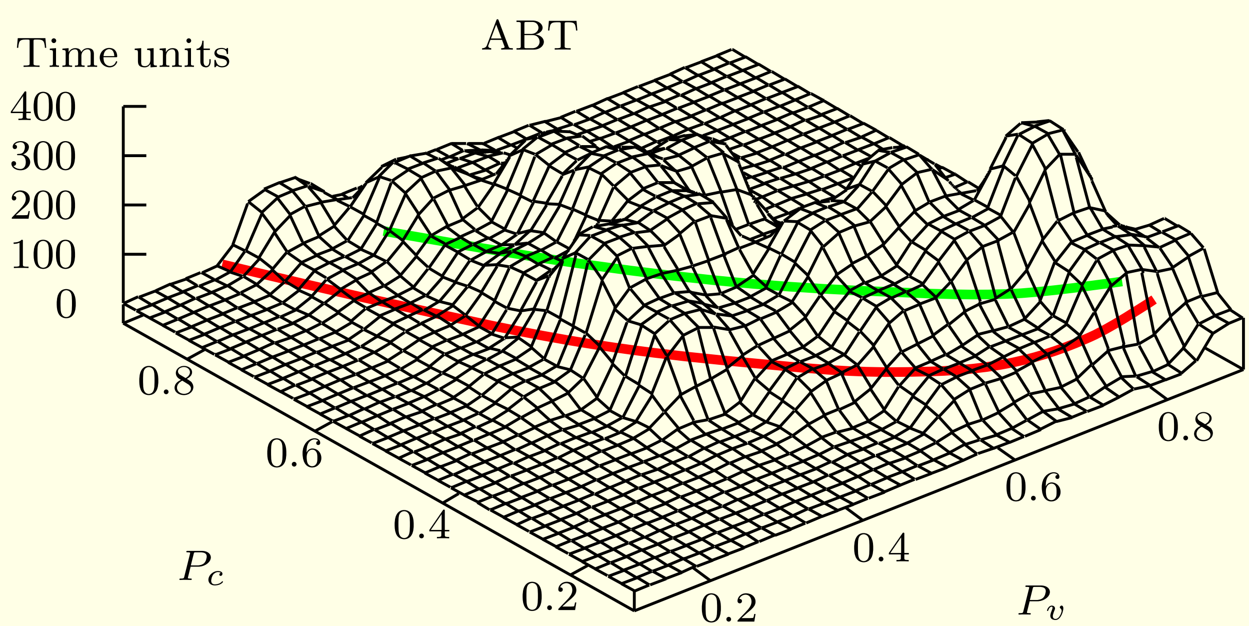


Mean solution time



$P_{sat} = 0.8$

$P_{sat} = 0.2$



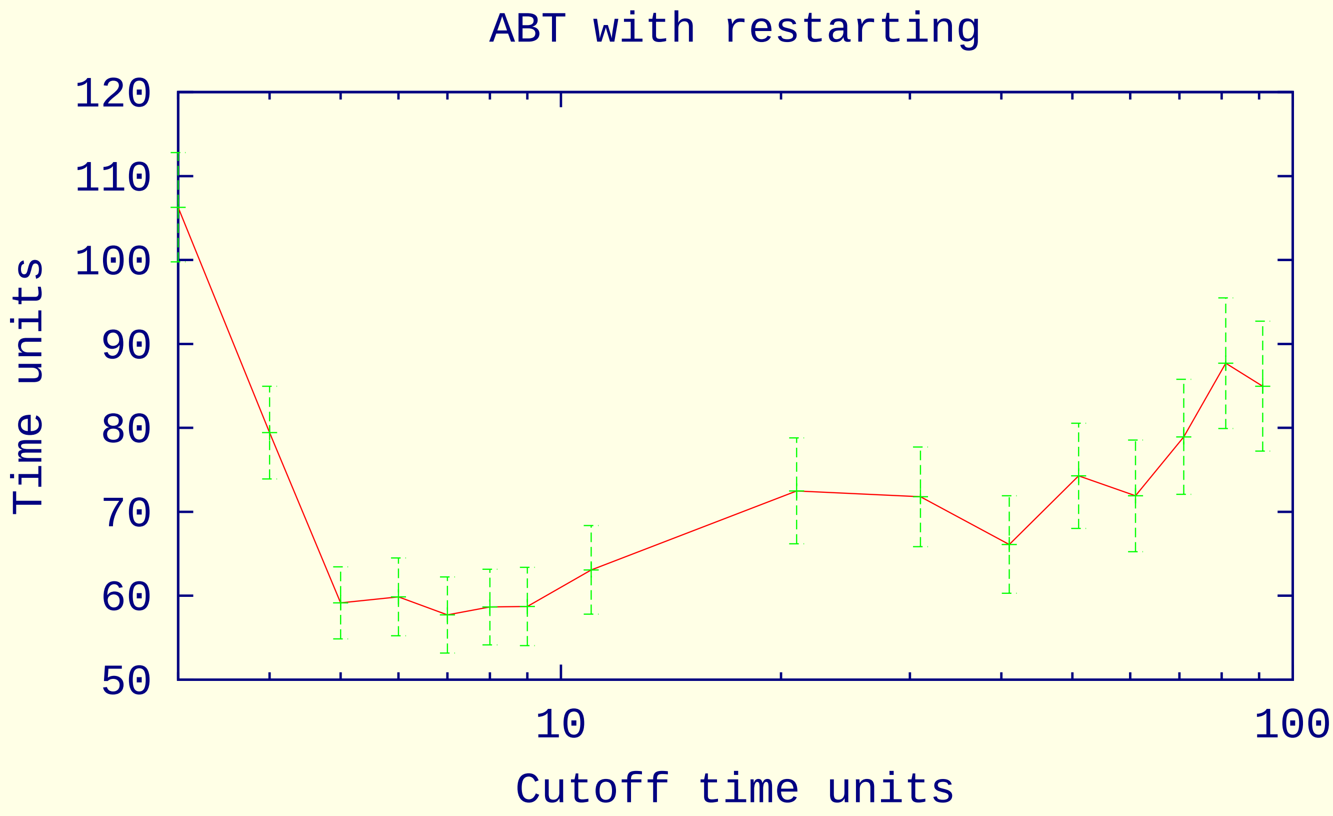
NP-complete for $P_c < 1$

harder for lower P_c

Restarting strategy for ABT

- Higher priority agent initiates the restarting of the search
- Performance depends on cutoff time
- Not suitable for AWC due to its inherent dynamic priority behavior

Mean solution time on a hard instance



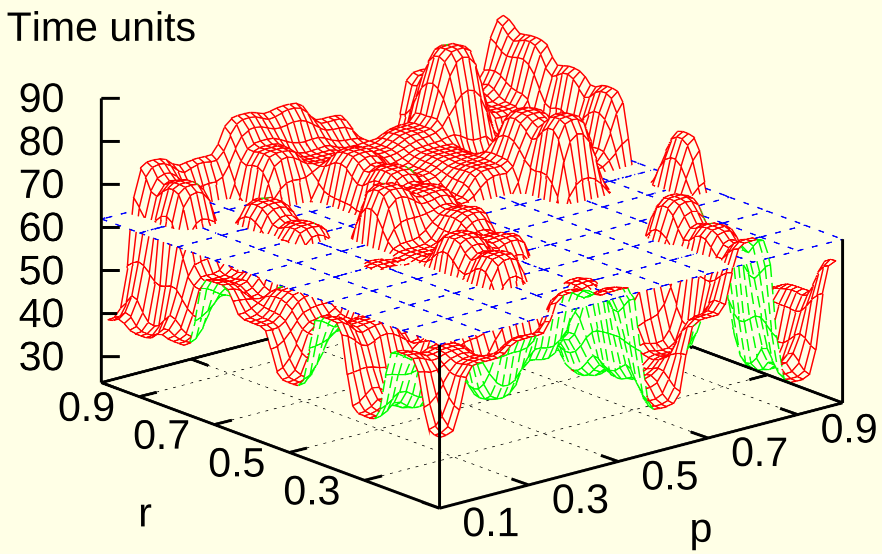
Active delaying of messages

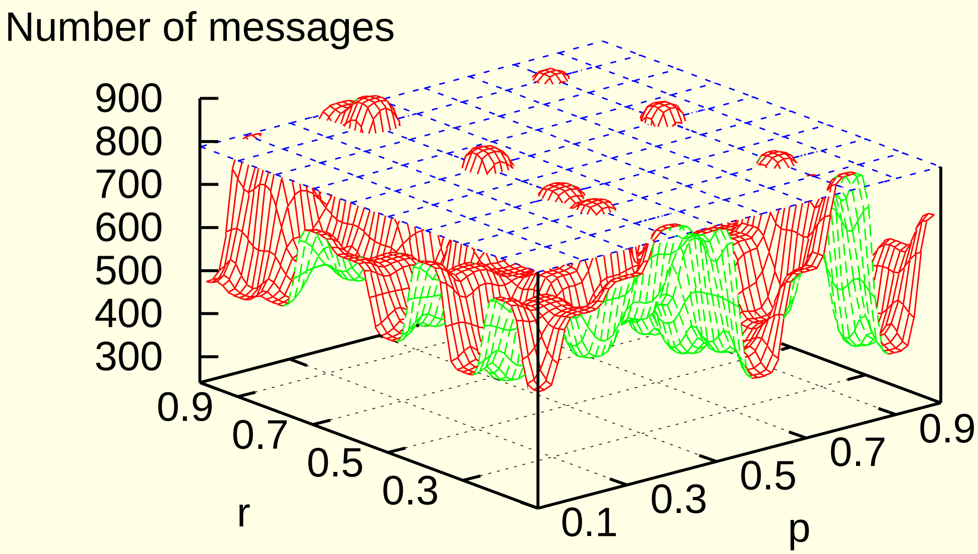
- Adding delay introduces a controlled source of randomization.

- Controlling the level of randomization:

  - p, probability that an agent adds extra delay
  - $0 \leq r \leq 1$, amount of added delay (fraction of the delay of the link)

Median values for a hard instance (AWC in a fixed delay scenario)



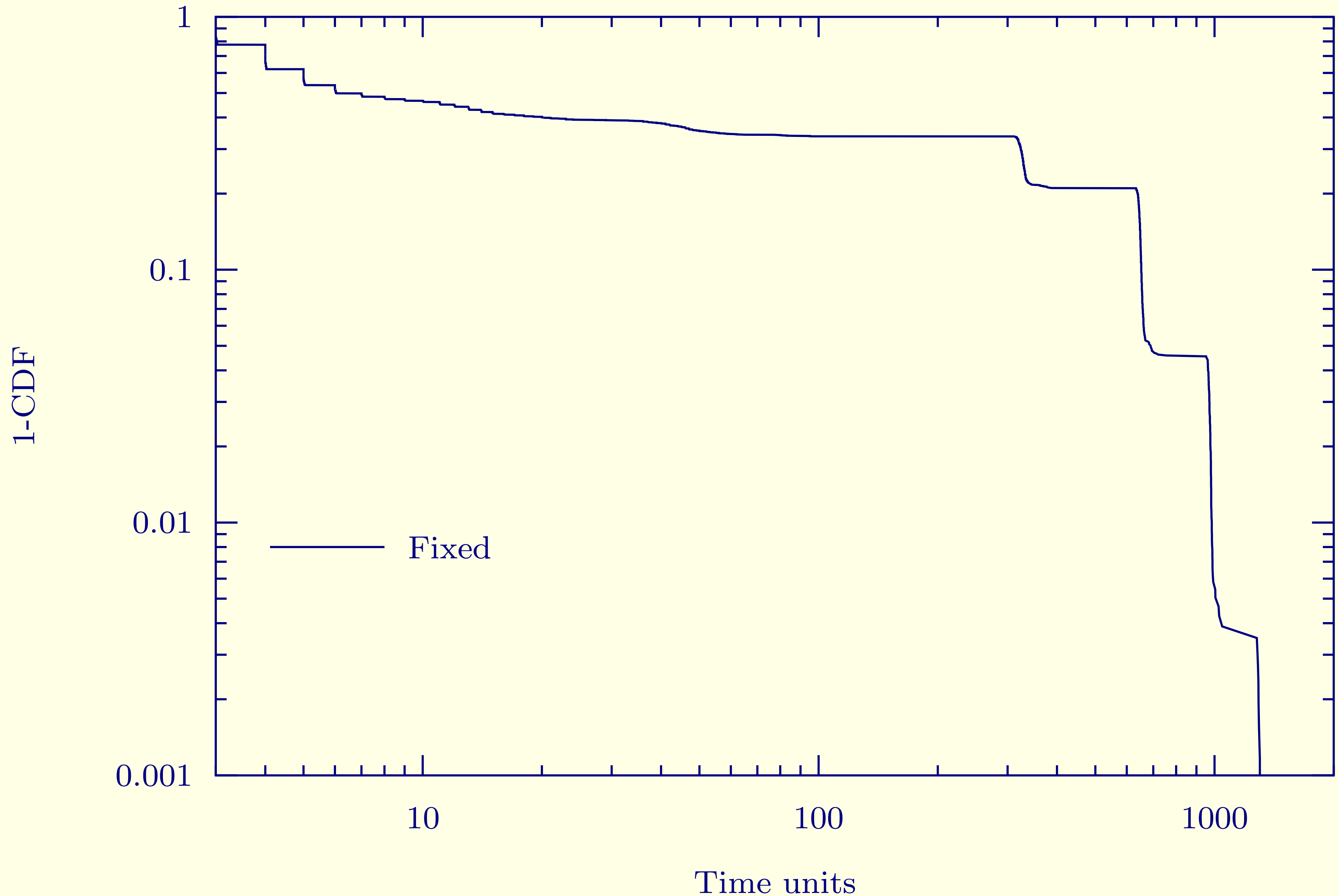


The effect of the communication network load

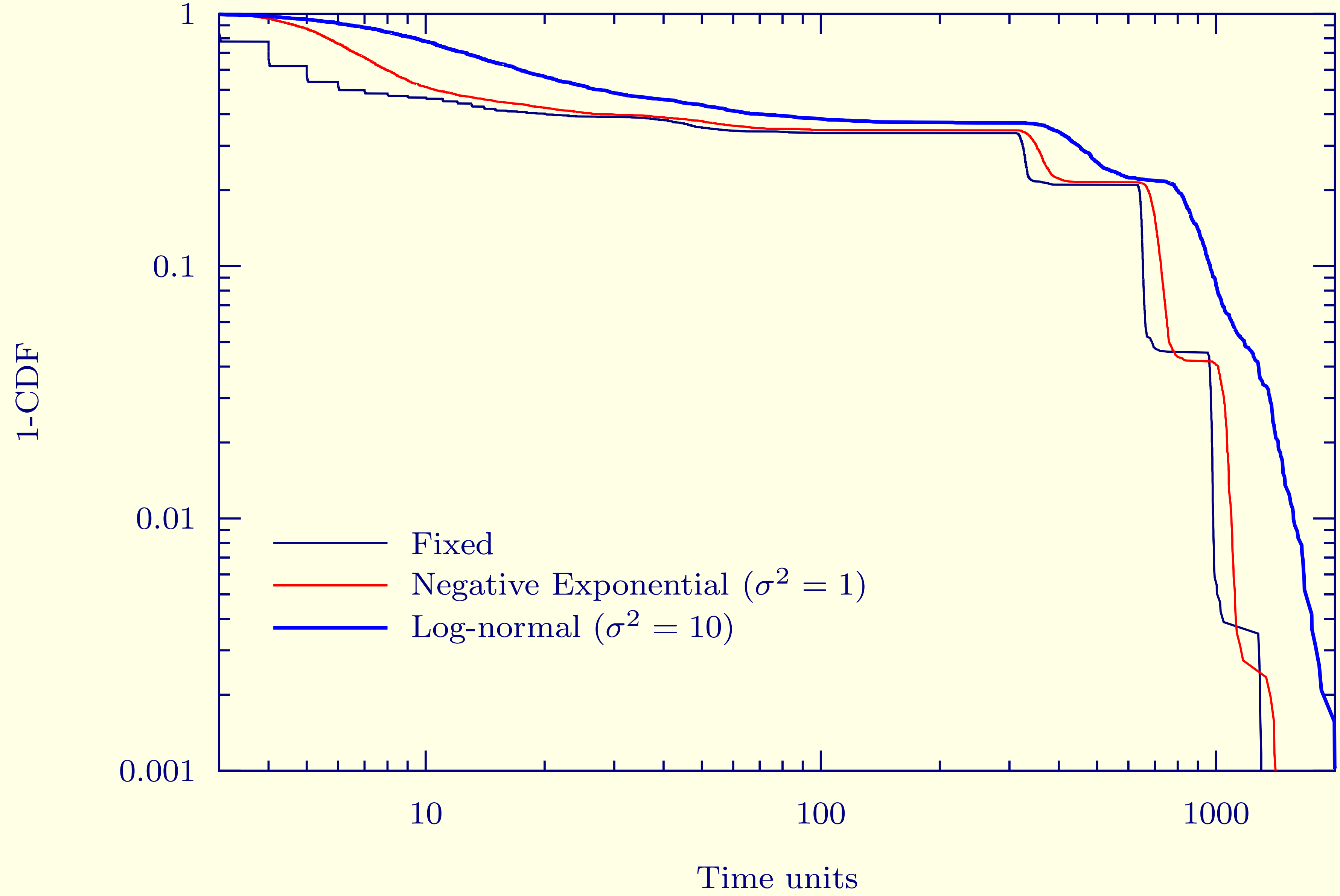
- Different traffic conditions modeled by different delay distributions

- Random delays impact algorithms performance in different ways:
  - **ABT:** Fixed delays better than random delays.
    The greater the variance of the delays,
    the worse the performance of ABT
  - **ABT with restarting:**
    Fairly robust to the variance of the delays
  - **AWC:** Random delays better than fixed delays.
    Extremely robust to the variance of the delays

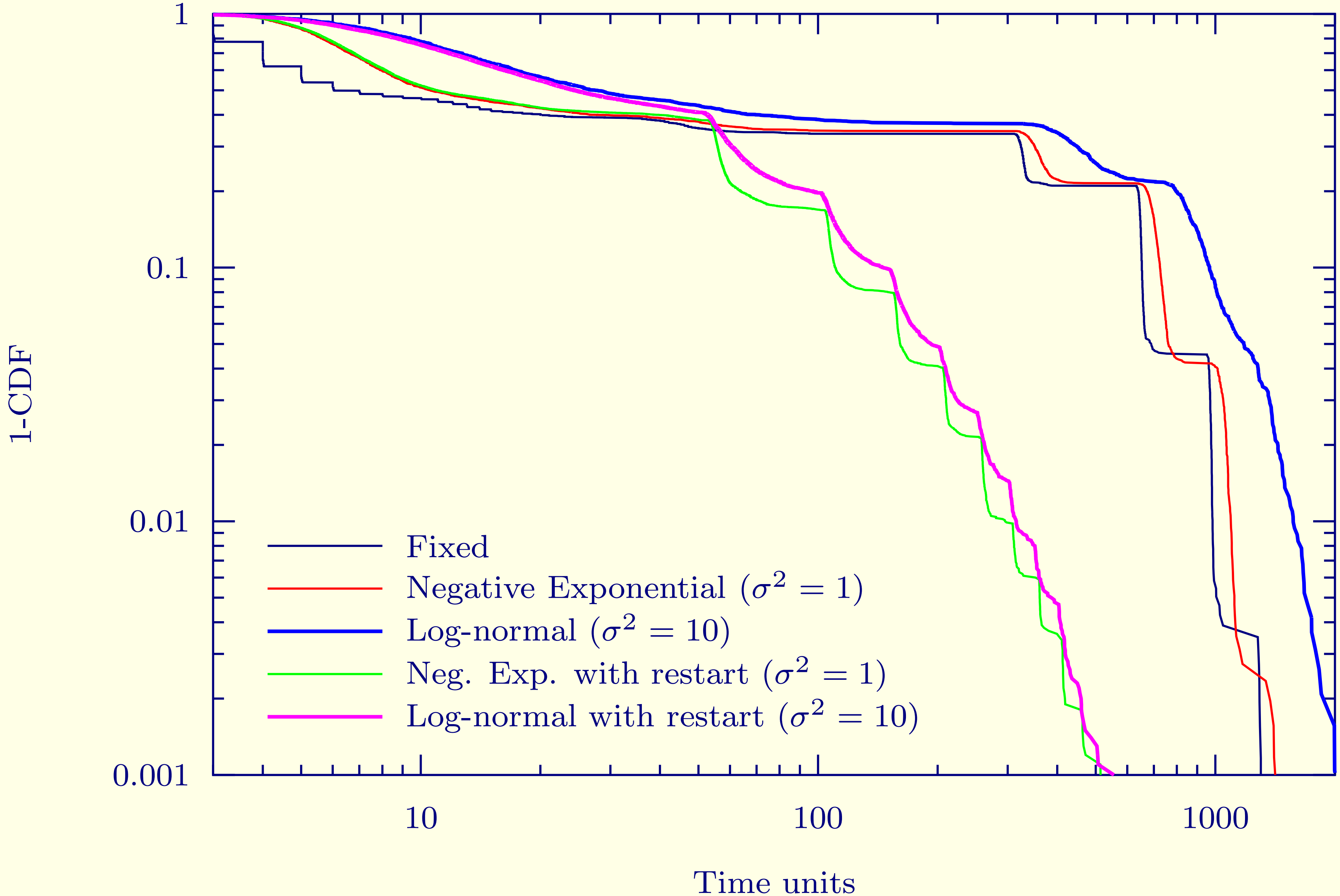
Cumulative density functions of time to solve a hard instance (ABT)



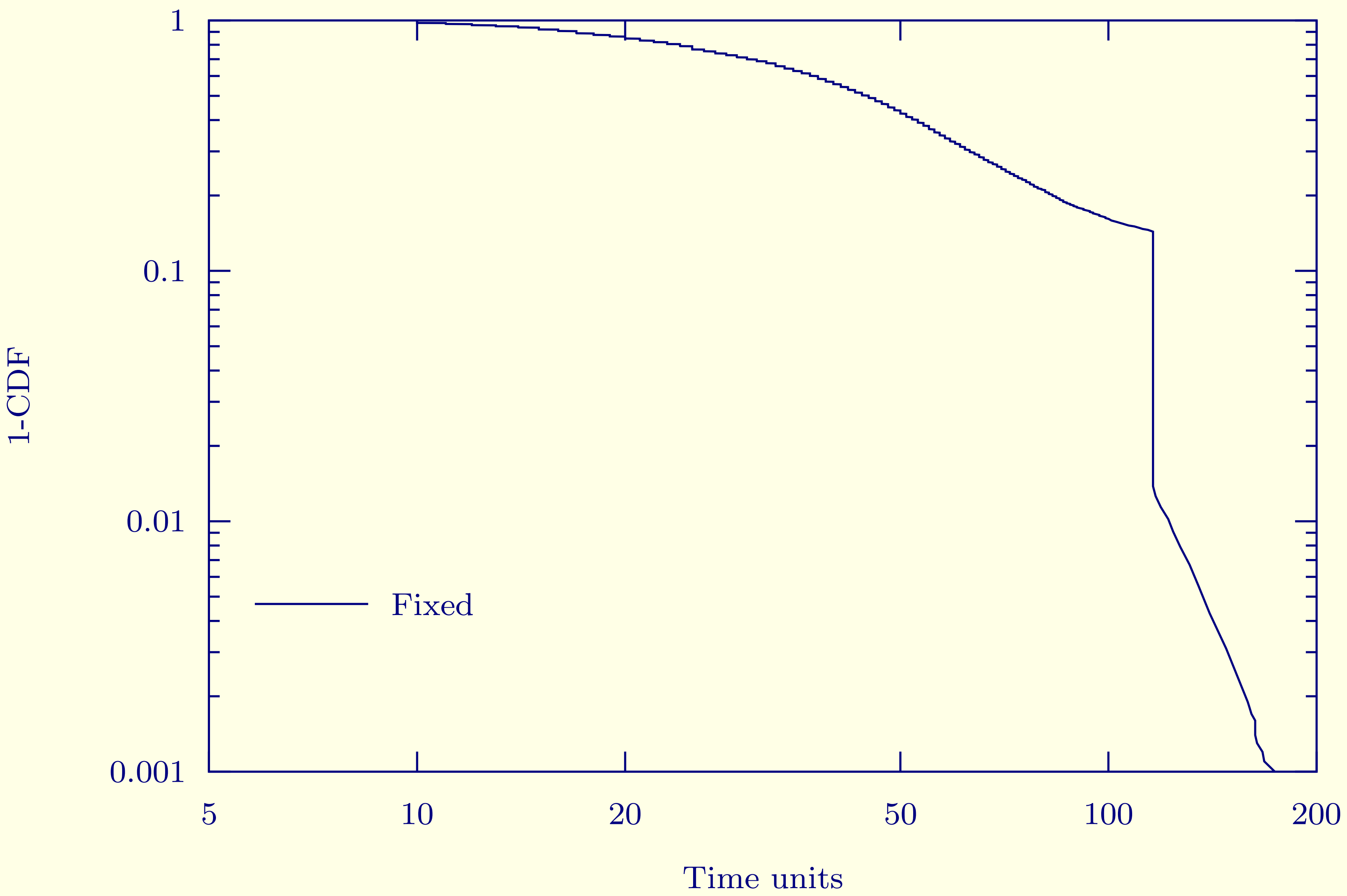
Cumulative density functions of time to solve a hard instance (ABT)



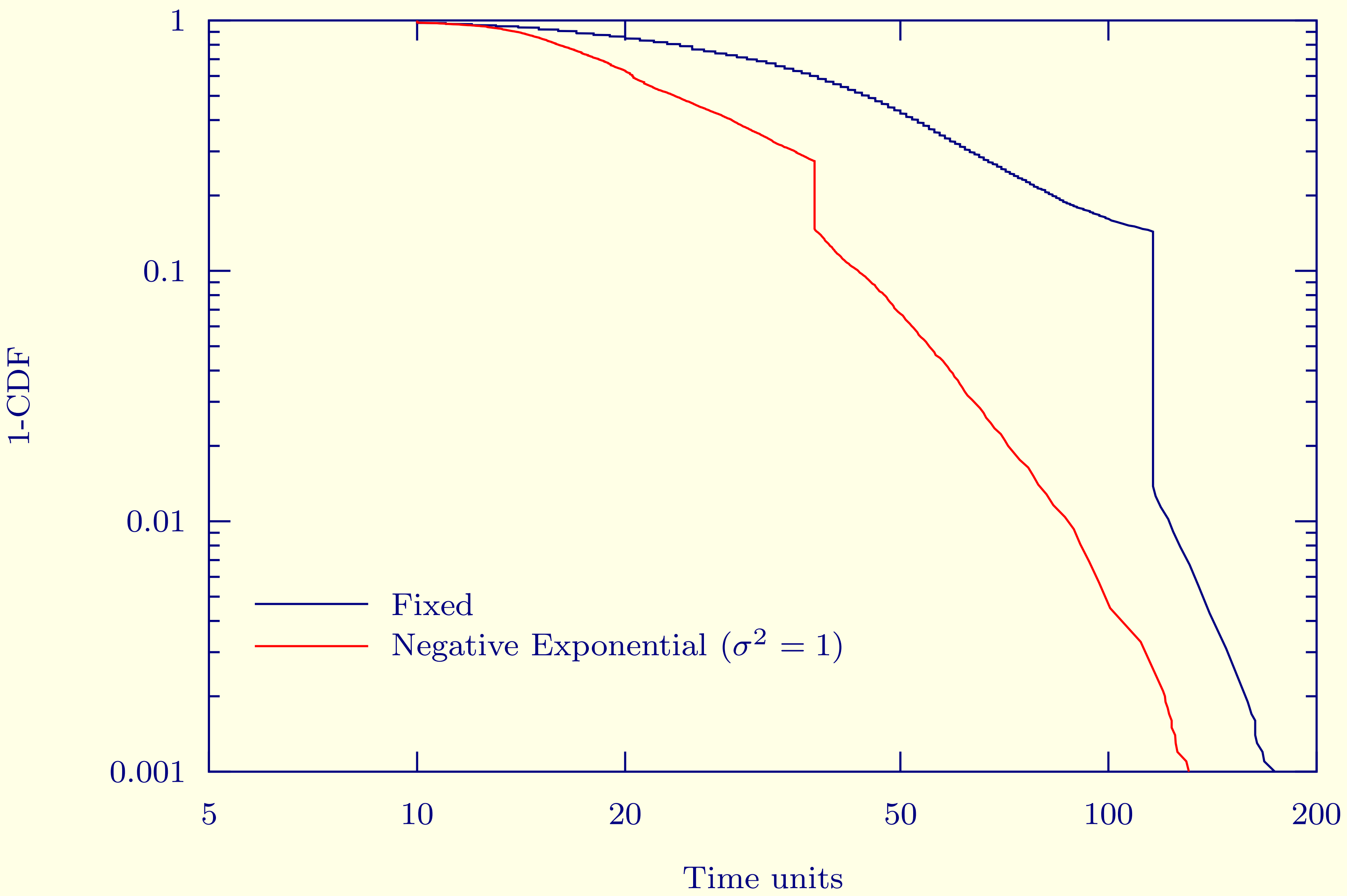
Cumulative density functions of time to solve a hard instance (ABT)



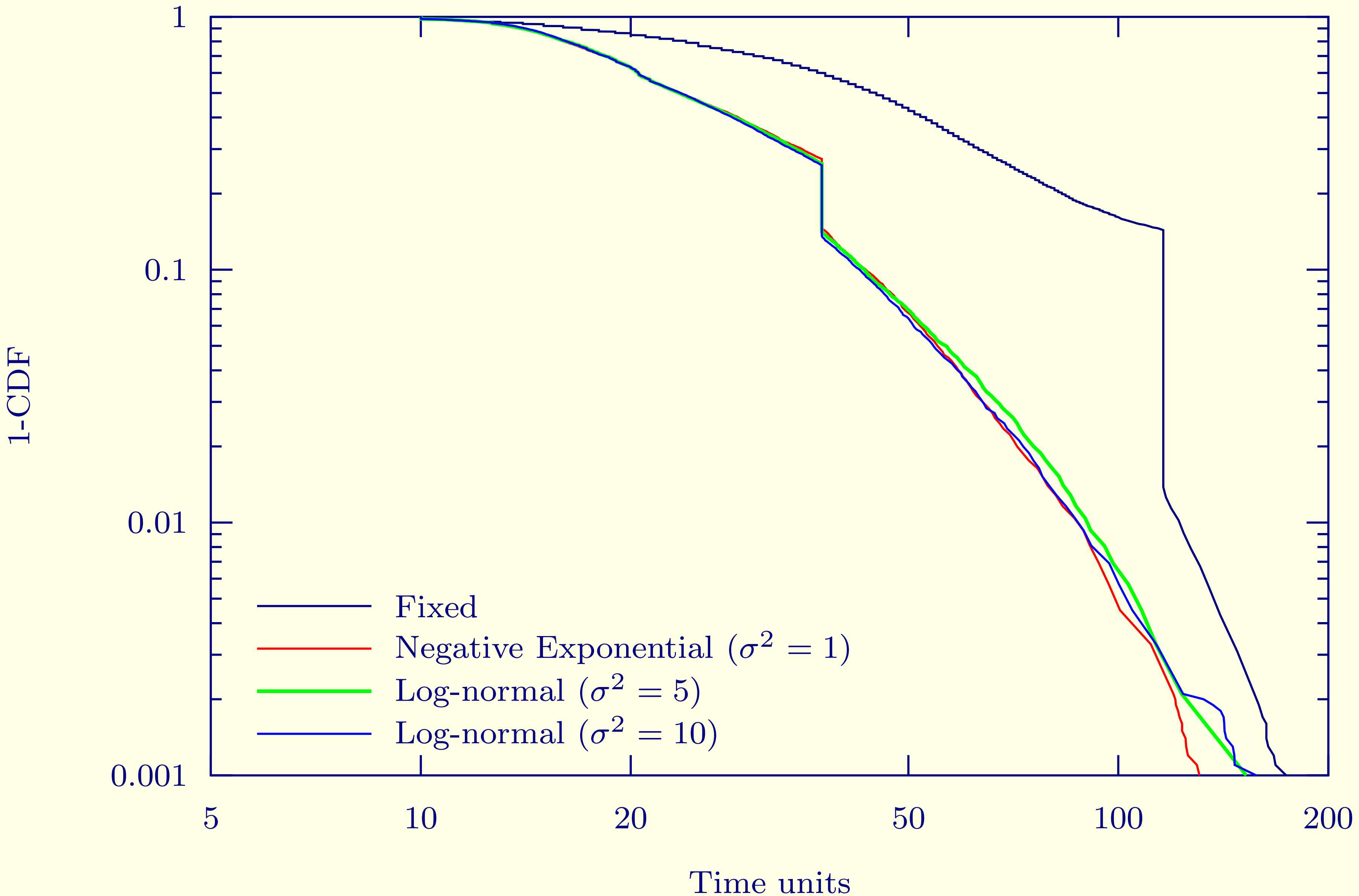
Cumulative density functions of time to solve a hard instance (AWC)



Cumulative density functions of time to solve a hard instance (AWC)



Cumulative density functions of time to solve a hard instance (AWC)



Conclusions

- Introduction of the SensorDCSP benchmark. Phase transition behavior in satisfiability. Constrainedness adjusted by the visibility and compatibility graphs
- Discrete-event simulator used to determine the impact of different network traffic conditions on the algorithms performance:
  - ABT performance worse for random delays
  - ABT improved using restarting
  - AWC performance better for random delays.
    Robustness in front of large random delay variations
  - AWC improved in fixed delay scenarios by the active addition of random delays

Conclusions

- Introduction of the SensorDCSP benchmark. Phase transition behavior in satisfiability. Constrainedness adjusted by the visibility and compatibility graphs
- Discrete-event simulator used to determine the impact of different network traffic conditions on the algorithms performance:
  - ABT performance worse for random delays
  - ABT improved using restarting
  - AWC performance better for random delays.
    Robustness in front of large random delay variations
  - AWC improved in fixed delay scenarios by the active addition of random delays

Conclusions

- Introduction of the SensorDCSP benchmark. Phase transition behavior in satisfiability. Constrainedness adjusted by the visibility and compatibility graphs
- Discrete-event simulator used to determine the impact of different network traffic conditions on the algorithms performance:
  - ABT performance worse for random delays
  - ABT improved using restarting
  - AWC performance better for random delays.
    Robustness in front of large random delay variations
  - AWC improved in fixed delay scenarios by the active addition of random delays