



Hierarchical Search on DisCSPs

Michael Orlov

`orlov@m@cs.bgu.ac.il`

Department of Computer Science
Ben-Gurion University

Advisor: Prof. Amnon Meisels

July 2006



- What is Constraint Satisfaction?
- What is Distributed CSP?
- Why do we use search?
- Why binary CSPs?

Introduction

Introduction

CSPs

Sudoku

DisCSPs

Motivation

Algorithms Overview

Group Partition

Distributed
Hierarchical Search

Descending
Requirements Search

Experimental Results

Beyond Search

References



Constraint Satisfaction Problems

Sudoku as a CSP

- n^4 variables in $n^2 \times n^2$ grid
- Domain of each variable: $\{1, \dots, n^2\}$, except for the “open” cells
- Constraints: values for each row, column, major $n \times n$ block are *alldifferent*

[Introduction](#)

[Introduction](#)

[CSPs](#)

[Sudoku](#)

[DisCSPs](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Constraint Satisfaction Problems

Sudoku as a CSP

			7	5				
	3			4	8		2	
1								6
	4							8
7	9						3	1
2							7	
5								7
	8		3	2			4	
				6	9			

[Introduction](#)

[Introduction](#)

[CSPs](#)

[Sudoku](#)

[DisCSPs](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



- ABT is the classical approach [Yokoo and Hirayama, 2000]
- Timetabling
- Privacy considerations
- Message delays
- Algorithm performance: CCs, messages [Meisels et al., 2002]

Introduction

Introduction

CSPs

Sudoku

DisCSPs

Motivation

Algorithms Overview

Group Partition

Distributed
Hierarchical Search

Descending
Requirements Search

Experimental Results

Beyond Search

References



Motivation

Introduction

Motivation

Motivation

PAs

Algorithms Overview

Group Partition

Distributed
Hierarchical Search

Descending
Requirements Search

Experimental Results

Beyond Search

References



- More parallelism and less backtracking
- Hierarchy facilitates smart PAs combination
- Independency of hierarchy and search

[Introduction](#)

[Motivation](#)

[Motivation](#)

[PAs](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Motivation: Combining PAs

- Theoretic motivation of DisHS
- *Constraint weight*: probability that a pair of values is not in conflict
- *Virtual constraint*: combination of constraints, weight approximated by multiplication
- Expected CCs when ordering k constraints by weight:

$$1 + w(k - 1) \leq E_k(w) \leq \frac{1 - w}{1 - \sqrt[k]{w}}$$

[Introduction](#)

[Motivation](#)

[Motivation](#)

[PAs](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithms Overview

Introduction

Motivation

Algorithms Overview

Hierarchical Search

Partition

DisHS

DesRS

Group Partition

Distributed
Hierarchical Search

Descending
Requirements Search

Experimental Results

Beyond Search

References



Hierarchical Search

- Partition
- Distributed Hierarchical Search
- Descending Requirements Search

Introduction

Motivation

Algorithms Overview

Hierarchical Search

Partition

DisHS

DesRS

Group Partition

Distributed
Hierarchical Search

Descending
Requirements Search

Experimental Results

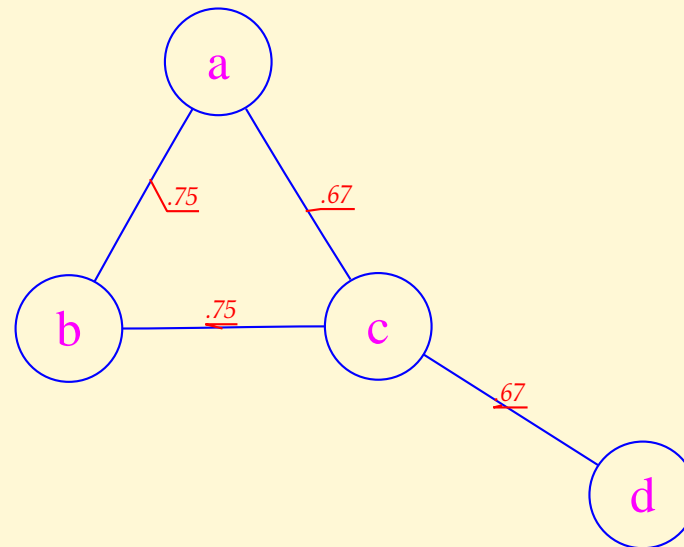
Beyond Search

References



Example: Partition

- Consider a small graph-coloring problem
- Domains of a , c and d are $\{1, 2, 3\}$
- Domain of b is $\{1, 2, 3, 4\}$



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

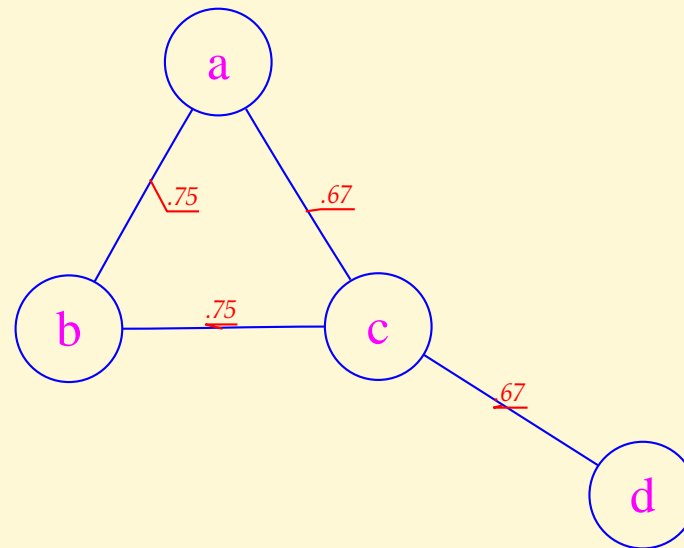
[Beyond Search](#)

[References](#)



Example: Partition

- Each agent sends Join to a minimal-weight neighbor
- $a \rightarrow c, b \rightarrow a, c \rightarrow d, d \rightarrow c$
- c and d join, send each other components info
- c sends Done to a, b



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

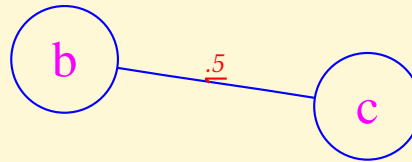
[Beyond Search](#)

[References](#)



Example: Partition

- a and b remove c from neighbors list
- a and b join, and send Done to c
- a, b, c, d send Leader messages to chosen leaders
- Leaders are activated at the next level (can't be confused by previous levels messages)



- b and c join, picking leader d

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

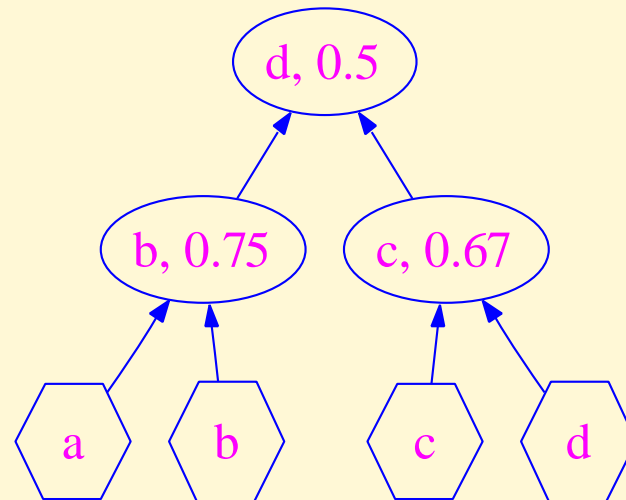
[Beyond Search](#)

[References](#)



Example: Partition

- We now have a hierarchy of agents
- *d* sends Search message to all agents in order to initiate the search process



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

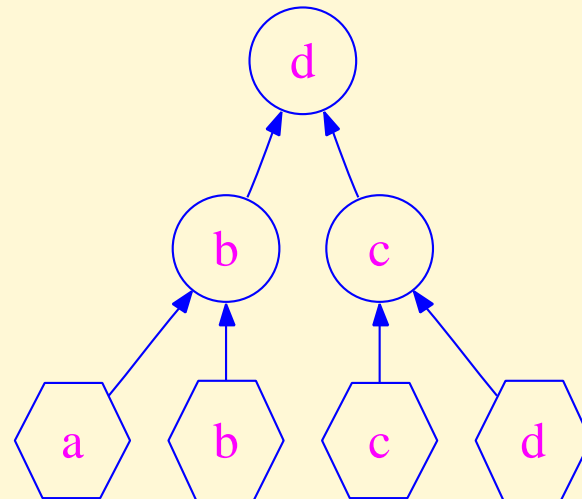
[Beyond Search](#)

[References](#)



Example: DisHS

- a, c, d send $\langle x = 1 \rangle, \langle x = 2 \rangle, \langle x = 3 \rangle$ to leaders b and c
- b sends $\langle x = 1 \rangle, \dots, \langle x = 4 \rangle$ to itself (actually, primitive agent to a leader)
- b, c prune inconsistent pairs and send results to d



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Example: DisHS

b sends consistent pairs to d :

a	b
1	2
1	3
1	4
2	1
2	3
2	4
3	1
3	2
3	4

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Example: DisHS

c sends consistent pairs to d :

c	d
1	2
1	3
2	1
2	3
3	1
3	2

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Example: DisHS

d prunes inconsistencies using Check queries to a, b, c and produces solutions

a	b	c	d
1	2	3	1
1	2	3	2
1	3	2	1
1	3	2	3
1	4	2	1
1	4	2	3
...			

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Example: DisHS

d prunes inconsistencies using Check queries to a, b, c and produces solutions

a	b	c	d
1	2	3	1
1	2	3	2
1	3	2	1
1	3	2	3
1	4	2	1
1	4	2	3
...			

Not necessary in this order!

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

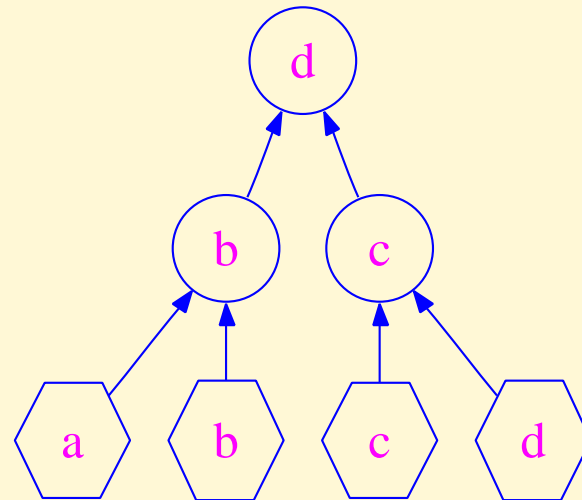
[Beyond Search](#)

[References](#)



Example: DesRS

- Each agent initiates a backtracking process
- Processes are independent, so let's consider one originating at *a*
- Note: numbers of processes and agents can be independent



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

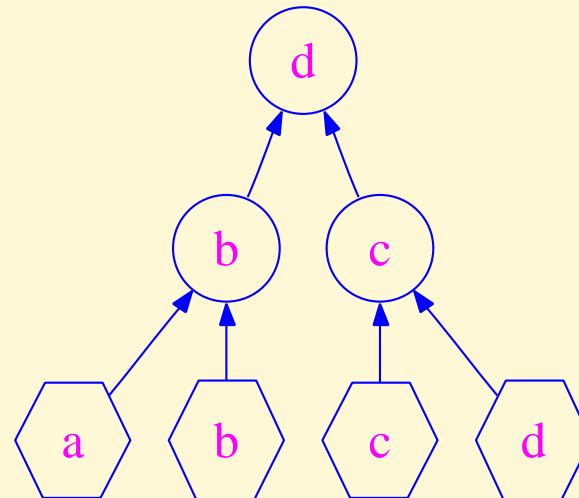
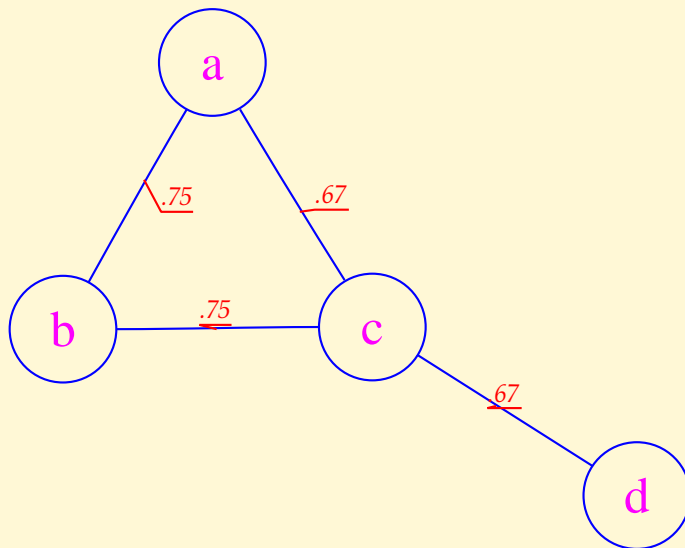
[Beyond Search](#)

[References](#)



Example: DesRS

- a to leader b : $\langle a = 1 \rangle$
- Leader b forwards to leaf b , which sends $\langle a = 1, b = 2 \rangle$ up
- Leader b forwards to leaf d via leaders d, c
- d to leader c : $\langle a = 1, b = 2, d = 1 \rangle$



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

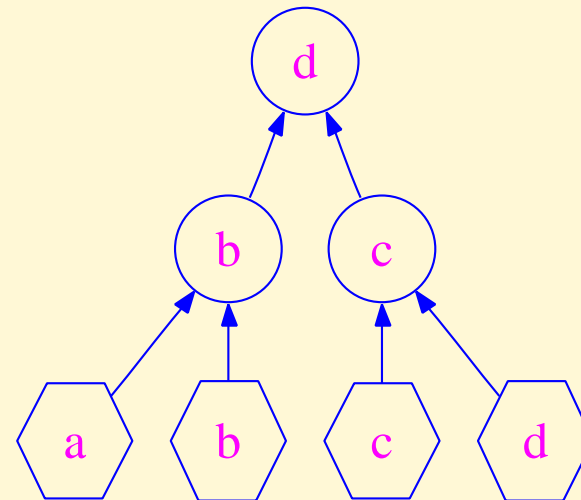
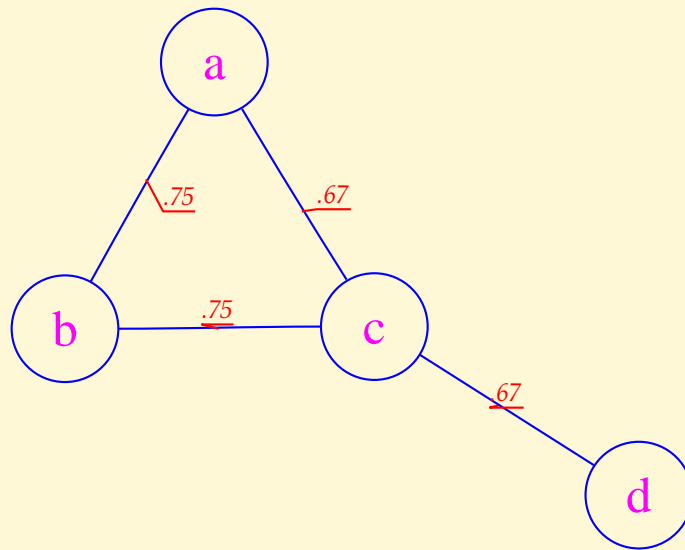
[Beyond Search](#)

[References](#)



Example: DesRS

- Leaf c sends to leader c a complete solution $\langle a = 1, b = 2, d = 1, c = 3 \rangle$
- Leader c forwards it to USER via leader d



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

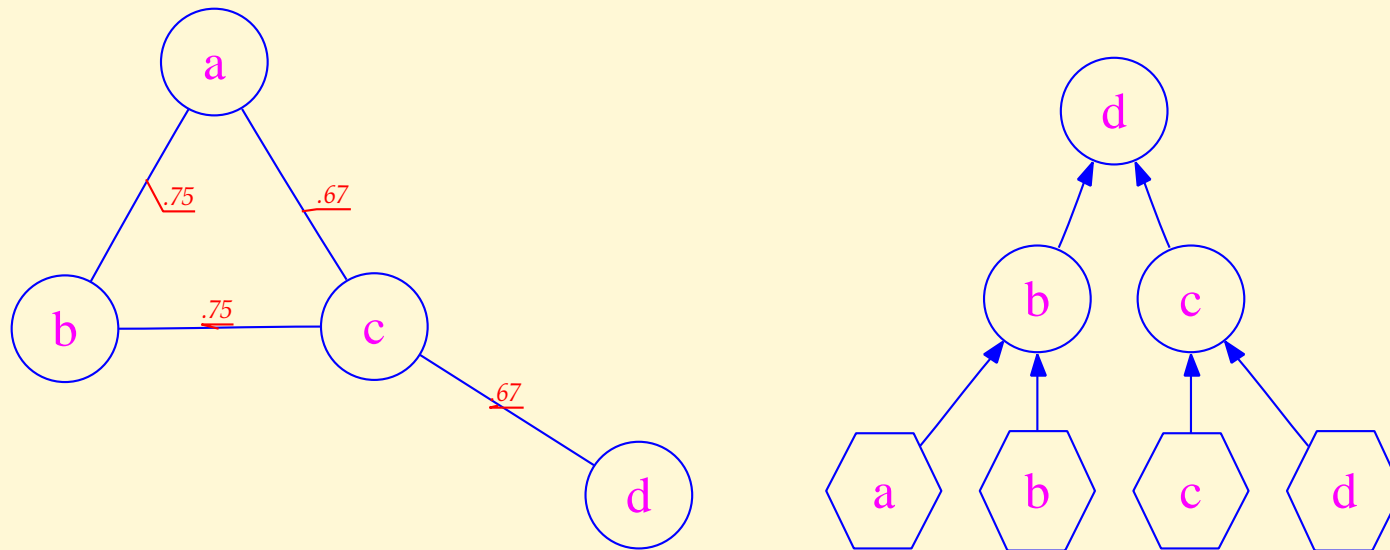
[Beyond Search](#)

[References](#)



Example: DesRS

- Leaf c sends to leader c a complete solution $\langle a = 1, b = 2, d = 1, c = 3 \rangle$
- Leader c forwards it to USER via leader d



No backtracking! But backtracking works as expected.

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Hierarchical Search](#)

[Partition](#)

[DisHS](#)

[DesRS](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Group Partition

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[**Group Partition**](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

- Agents try to join neighbors with minimal weight:
Join and NoJoin messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

- Agents try to join neighbors with minimal weight:
Join and NoJoin messages
- Joining agents exchange component information
with Components messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

- Agents try to join neighbors with minimal weight:
Join and NoJoin messages
- Joining agents exchange component information
with Components messages
- Removing joined neighbors and environment for
next level: Done messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

- Agents try to join neighbors with minimal weight: Join and NoJoin messages
- Joining agents exchange component information with Components messages
- Removing joined neighbors and environment for next level: Done messages
- Leader activation: Leader messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition Example](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

Input : agent s , its neighbors N

Output : components C , pairs, leader, parent \leftarrow USER

Locals : \tilde{N} , N_{ldr} , g , level $\leftarrow p$, start \leftarrow TRUE, next-level

SEND(s , Leader(N , $\{(s, N, 0)\}$, TRUE, 0))

loop forever do

switch RECEIVE() **do**

 ...

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

```
case Join⟨t⟩
  if t = g then
    SEND(g, Components⟨C⟩)
  else
    SEND(t, NoJoin)
case NoJoin
  g ← SELECT(Ñ)
  SEND(g, Join⟨s⟩)
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

```
case Components  $\langle C_g \rangle$ 
  if  $s \neq g$  then
     $\{ \text{leader}, C \} \leftarrow \text{SELECT-LEADER}(C, C_g, \text{next-level})$ 
    if parent = USER then
      parent  $\leftarrow$  leader
  else
    leader  $\leftarrow$  s
  foreach  $t \in N \cup \{s\}$  do
    SEND( $t$ , Done  $\langle s, \text{leader} \rangle$ )
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

```
case Done⟨t, leadert⟩
  if t ≠ s then
    Nldr ← UPDATE(Nldr, N, t, leadert)
    Ñ ← Ñ ∖ {t}
  if Ñ = NIL then
    level ← p
    SEND(leader, Leader⟨Nldr ∖ {leader}, C, s = g,
      next-level⟩)
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

```
case Leader⟨N', C', single, level'⟩
  if start then
    N ← N', C ← C', Nldr ← NIL
  else
    pairs ← {({t,r}, w) : (t,  $\hat{N}$ ) ∈ C ∧ r ∈ C' ∧ (r, w) ∈  $\hat{N}$ }
    N ← COMBINE(N, N'), C ← C ∪ C'
  start ← ¬start ∨ single
  ...
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: Partition

```
case Leader⟨N', C', single, level'⟩
  ...
  if start then
    if N = NIL then
      leader ← USER
      foreach t ∈ C do
        SEND(t, Search)
    else
      level ← level', next-level ← level + 1
      Ñ ← N ∪ {(s, 1.5)}
      g ← SELECT(Ñ)
      SEND(g, Join⟨s⟩)
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed](#)

[Hierarchical Search](#)

[Descending](#)

[Requirements Search](#)

[Experimental Results](#)

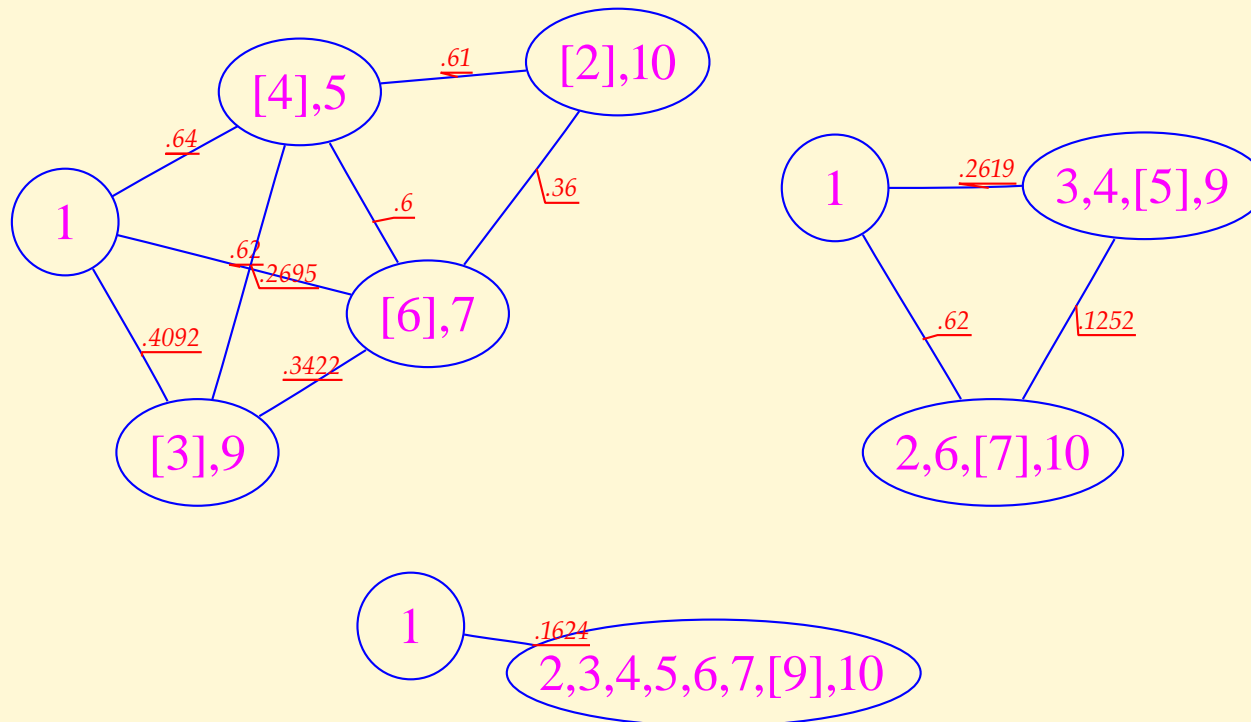
[Beyond Search](#)

[References](#)



Algorithm: Partition (Example)

■ Subsequent levels during partitioning



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

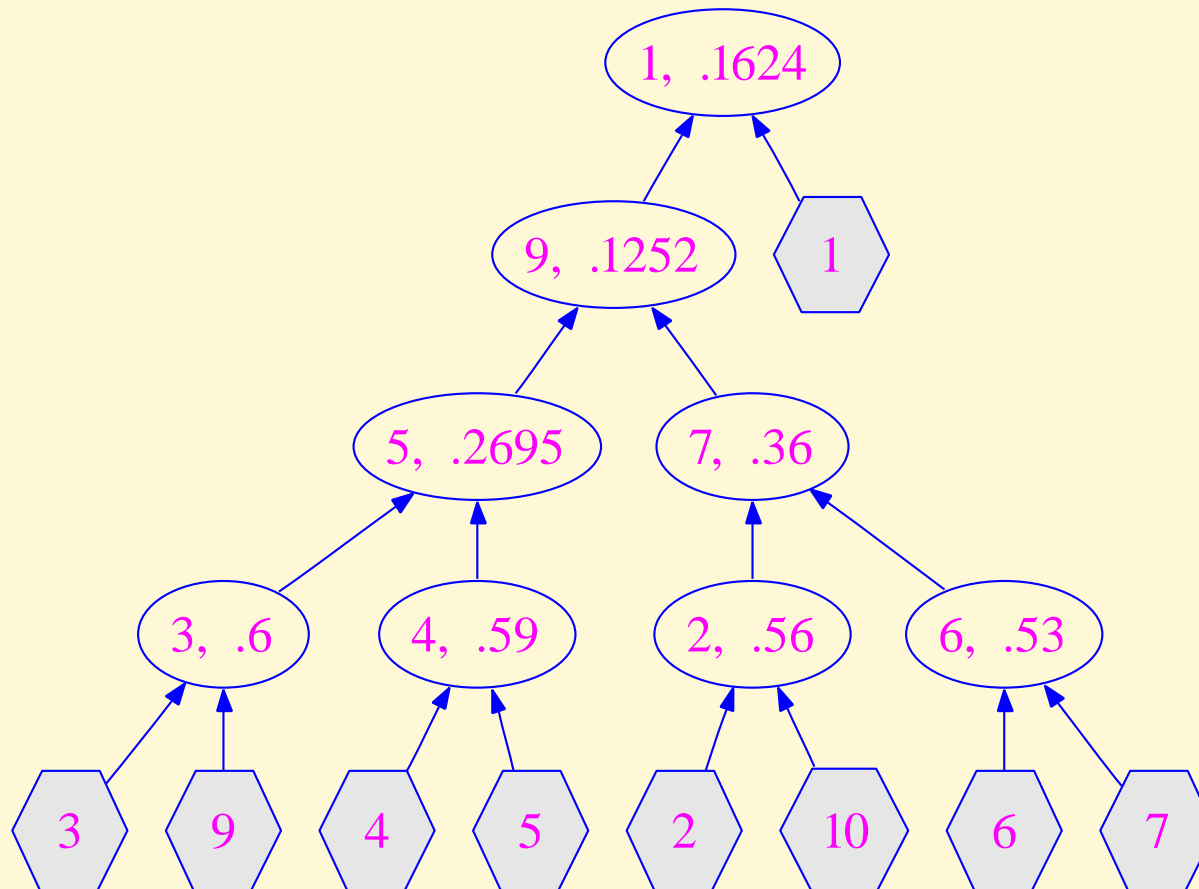
[Beyond Search](#)

[References](#)



Algorithm: Partition (Example)

■ Resulting hierarchy



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Partition](#)

[Algorithm: Partition](#)

[Example](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Distributed Hierarchical Search

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[**Distributed
Hierarchical Search**](#)

[DisHS](#)

[Algorithm: DisHS](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

- Agents send consistent PAs up in the hierarchy with Assignment messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[**DisHS**](#)

[Algorithm: DisHS](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

- Agents send consistent PAs up in the hierarchy with Assignment messages
- Representative agents combine PAs

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[**DisHS**](#)

[Algorithm: DisHS](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

- Agents send consistent PAs up in the hierarchy with Assignment messages
- Representative agents combine PAs
- Check and Answer messages are used for value compatibility queries

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[**DisHS**](#)

[Algorithm: DisHS](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

- Agents send consistent PAs up in the hierarchy with Assignment messages
- Representative agents combine PAs
- Check and Answer messages are used for value compatibility queries
- **Problem:** message queues saturation

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[**DisHS**](#)

[Algorithm: DisHS](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

- Agents send consistent PAs up in the hierarchy with Assignment messages
- Representative agents combine PAs
- Check and Answer messages are used for value compatibility queries
- **Problem:** message queues saturation
- **Extensions:** on-demand assignments, query caching, message priority

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[DisHS](#)

[Algorithm: DisHS](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

Input : agent s , partition output, domain D

Output : a global solution is sent to USER

Locals : $\text{row} \leftarrow 0$, $\text{Pending}[\cdot] \leftarrow 0$, $\text{Solutions}[\cdot]$,
 $\text{Iterator} \leftarrow \text{NIL}$, $\text{requests} \leftarrow 1$

loop forever do

switch RECEIVE() **do**

case Search

$\text{level} \leftarrow \xi$

forall $v \in D$ **do**

 SEND(parent, Assignment $\langle s, \{\langle s, v \rangle\} \rangle$)

 SEND(parent, Assignment $\langle s, \text{STOP} \rangle$)

 ...

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[DisHS](#)

[Algorithm: DisHS](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

```
case Check⟨t, row', {⟨s, v⟩, ⟨r, w⟩}⟩  
    SEND(t, Answer⟨row', CHECK(v, r, w)⟩)  
case Answer⟨row', ok⟩  
    if Pending[row'] ≠ 0 then  
        if ¬ok then  
            Pending[row'] ← 0  
            requests ← requests + 1  
            PROCESS-REQUEST()  
        else  
            Pending[row'] ← Pending[row'] - 1  
            if Pending[row'] = 0 then  
                SEND(leader, Assignment⟨s, Solutions[row']⟩)
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[DisHS](#)

[Algorithm: DisHS](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DisHS

case Request

requests \leftarrow requests + 1

PROCESS-REQUEST()

case Assignment(t, partial)

ITERATOR-ADD(iterator, t, partial)

PROCESS-REQUEST()

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[DisHS](#)

[**Algorithm: DisHS**](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Descending Requirements Search

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

**[Descending
Requirements Search](#)**

[DesRS](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

- Each agent initiates a backtracking search

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[**DesRS**](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

- Each agent initiates a backtracking search
- Representative agents serve as routers

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[**DesRS**](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

- Each agent initiates a backtracking search
- Representative agents serve as routers
- Search processes are independent

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[**DesRS**](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

- Each agent initiates a backtracking search
- Representative agents serve as routers
- Search processes are independent
- Assignment and Nogood messages

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[**DesRS**](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

Input : agent s , partition output, domain D , child agents

$c_{0,1}$, primitive child indicators $\text{prim}_{0,1}$

Output : a global solution is sent to USER

Locals : Id-Map[.]

loop forever do

switch RECEIVE() **do**

case Search

 level $\leftarrow \xi$

 SEND(s , Assignment($\langle s, s, \text{NIL}, \text{TRUE} \rangle$))

 ...

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[DesRS](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

```
case Assignment⟨t, id, PA, primitive⟩
  if primitive then
    Id-Map[id] ← ⟨PA, D, ∅⟩
  else if ∃i : t = ci then
    if c1-i ∈ PA then
      SEND(leader, Assignment⟨s, id, PA, FALSE⟩)
    else
      SEND(c1-i, Assignment⟨s, id, PA, prim1-i⟩)
  else
    i ← RANDOM({0,1})
    SEND(ci, Assignment⟨s, id, PA, primi⟩)
case Nogood⟨id, exp⟩
  ⟨·, ·, united-exp⟩ ← Id-Map[id]
  united-exp ← united-exp ∪ exp
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[DesRS](#)

Algorithm: DesRS

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

```
case Assignment⟨·, id, ·, TRUE⟩ ∨ Nogood⟨id, ·⟩
  ⟨PA, Values, exp⟩ ← Id-Map[id]
  v ← NIL
  while v = NIL ∧ Values ≠ ∅ do
    v ← RANDOM(Values)
    Values ← Values ∖ {v}
    for (r = w) ∈ PA (left-to-right, neighbors of s only) do
      if CHECK(v, r, w) then
        exp ← exp ∪ {r}
        v ← NIL
      break
  ...
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[DesRS](#)

[Algorithm: DesRS](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Algorithm: DesRS

```
case Assignment⟨·, id, ·, TRUE⟩ ∨ Nogood⟨id, ·⟩
...
if  $v \neq \text{NIL}$  then
    SEND(parent, Assignment⟨s, id, ⟨PA, (s = v)⟩, FALSE⟩)
else if  $\text{exp} \neq \emptyset$  then
    for  $r \in \text{PA}$  (right-to-left) do
        if  $r \in \text{exp}$  then
            SEND(r, Nogood⟨id,  $\text{exp} \setminus \{r\}$ ⟩)
            break
else
    SEND(USER, Nogood⟨id, exp⟩)
```

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[DesRS](#)

Algorithm: DesRS

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Experimental Results

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[**Experimental Results**](#)

DisHS

DesRS, ABT

DesRS, ConcDB

AntiDisHS

AntiDesRS

[Beyond Search](#)

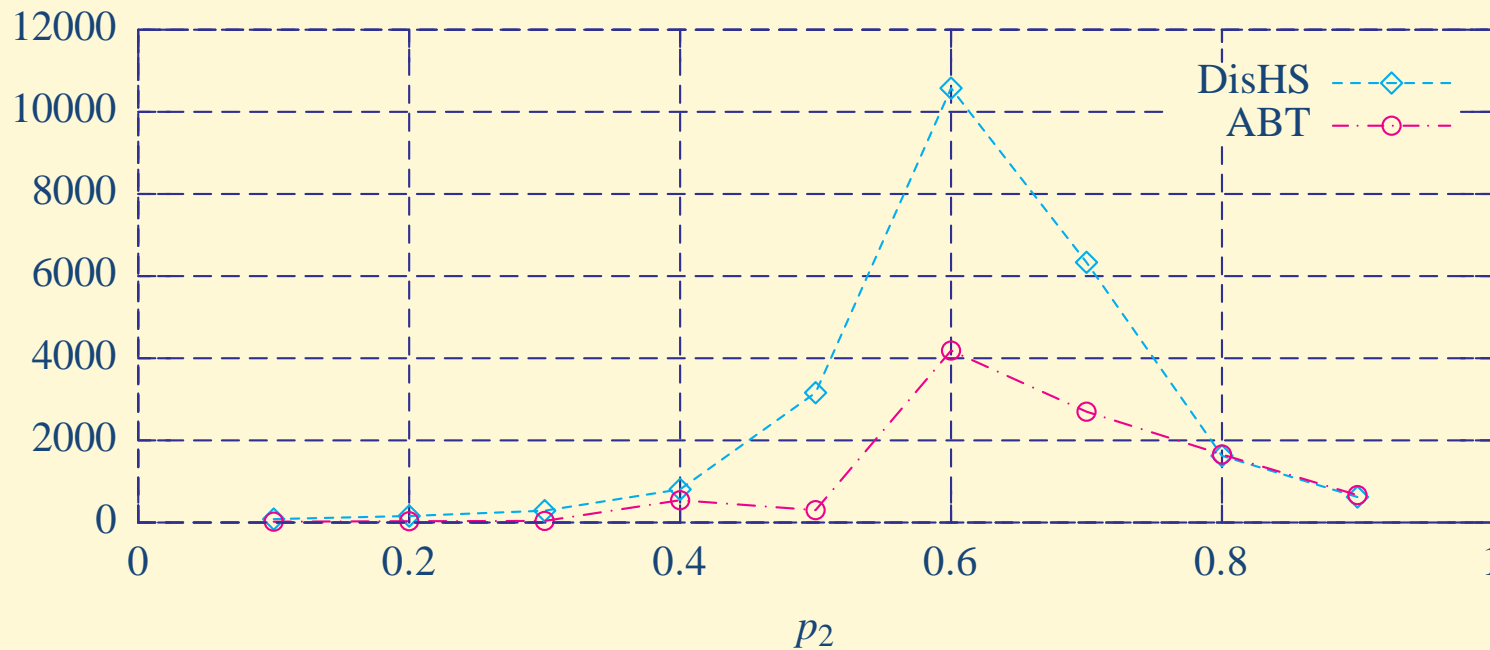
[References](#)



Experimental Results: DisHS

- DisHS, ABT on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$

Constraint checks: DisHS and ABT



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

DisHS

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

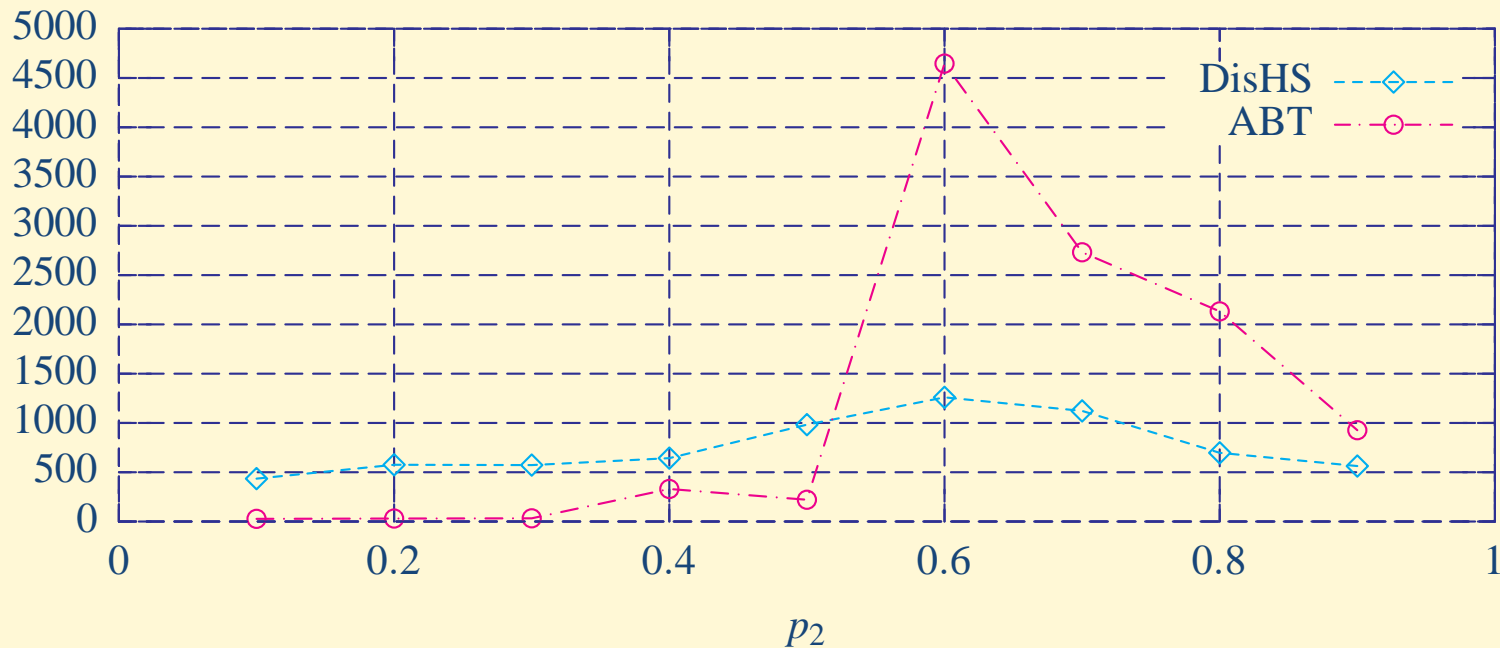
[References](#)



Experimental Results: DisHS

- DisHS, ABT on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$

Messages: DisHS and ABT



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

DisHS

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

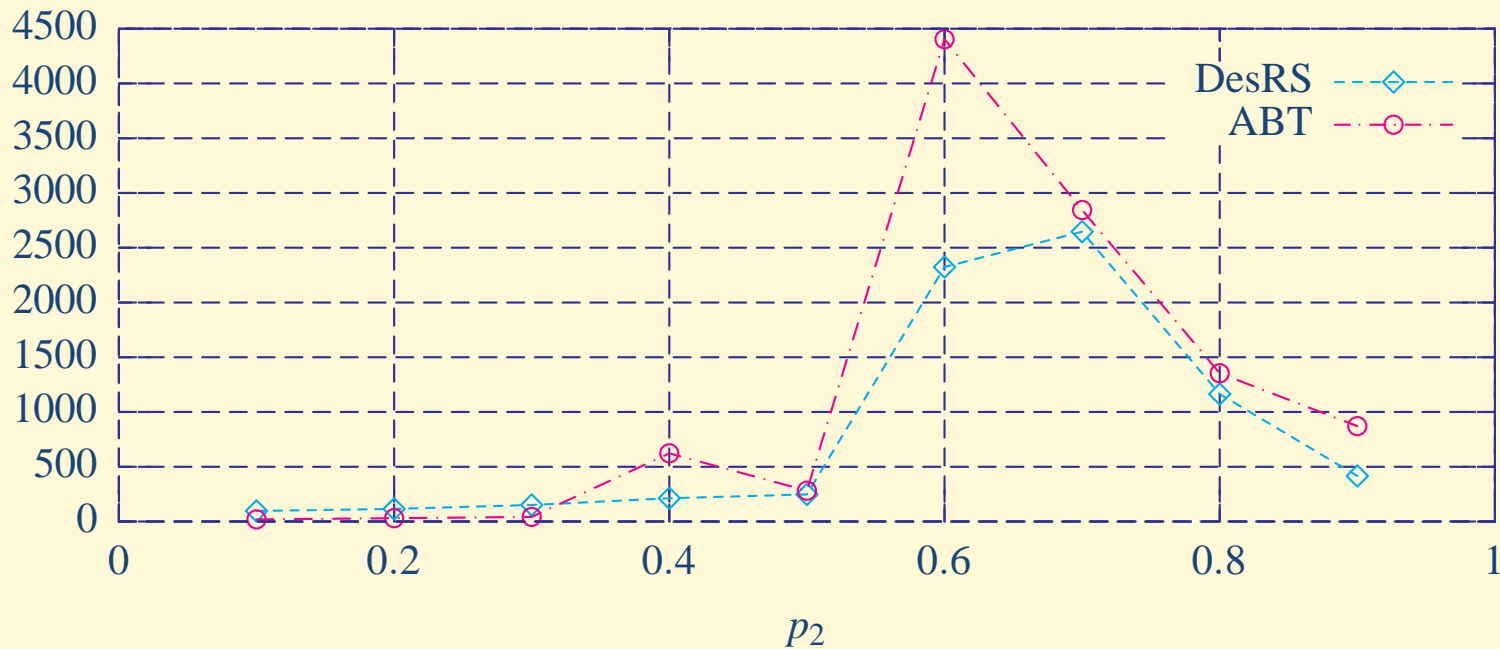
[References](#)



Experimental Results: DesRS

- DesRS, ABT on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$

Constraint checks: DesRS and ABT



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

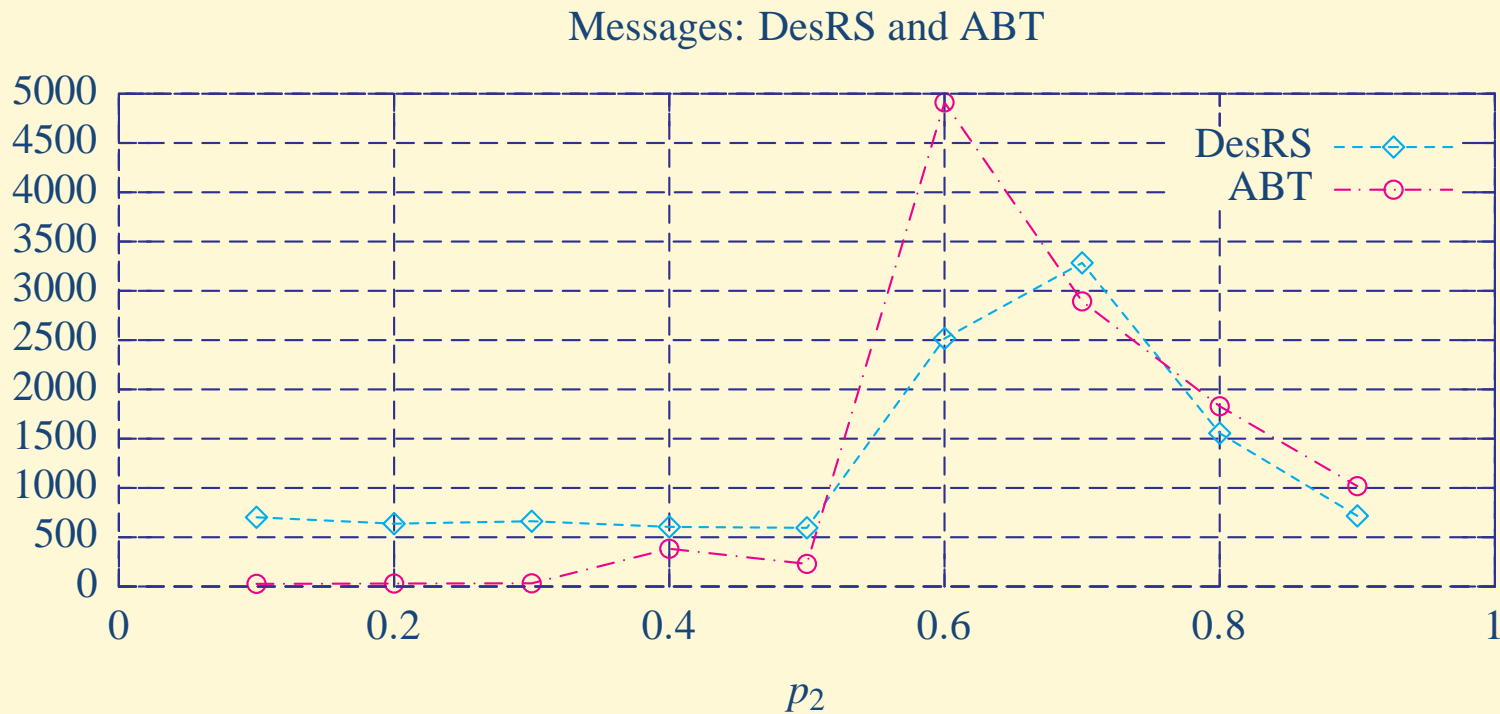
[Beyond Search](#)

[References](#)



Experimental Results: DesRS

- DesRS, ABT on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

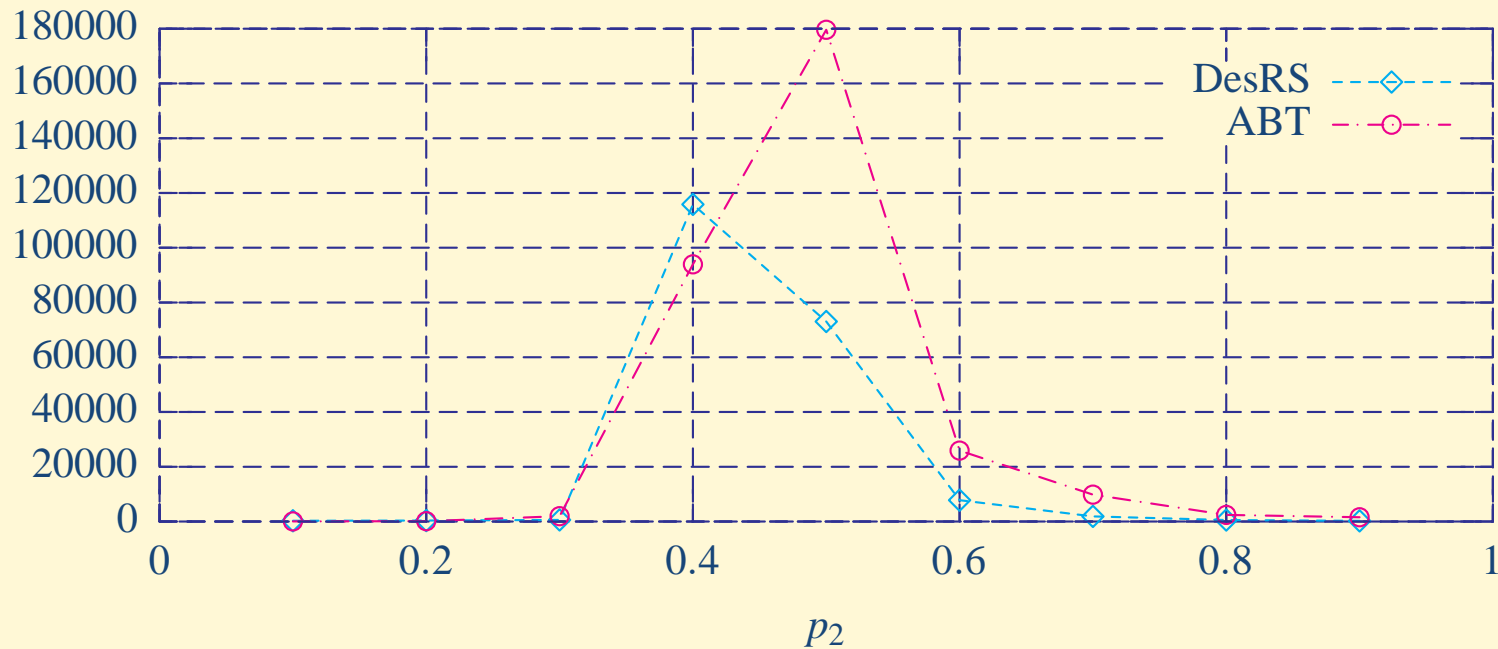
[References](#)



Experimental Results: DesRS

- DesRS, ABT on random problems with 20 agents, domain size of 10, and $p_1 = 0.4$

Constraint checks: DesRS and ABT



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

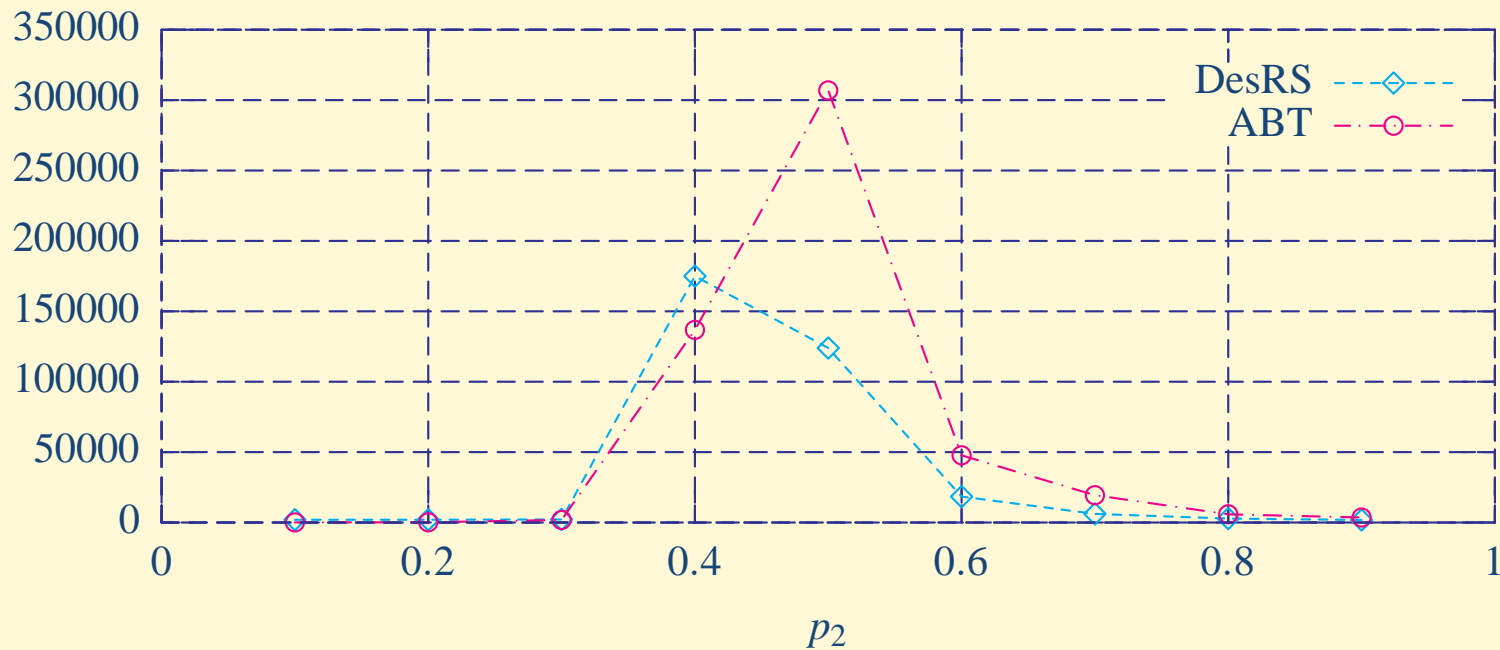
[References](#)



Experimental Results: DesRS

- DesRS, ABT on random problems with 20 agents, domain size of 10, and $p_1 = 0.4$

Messages: DesRS and ABT



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

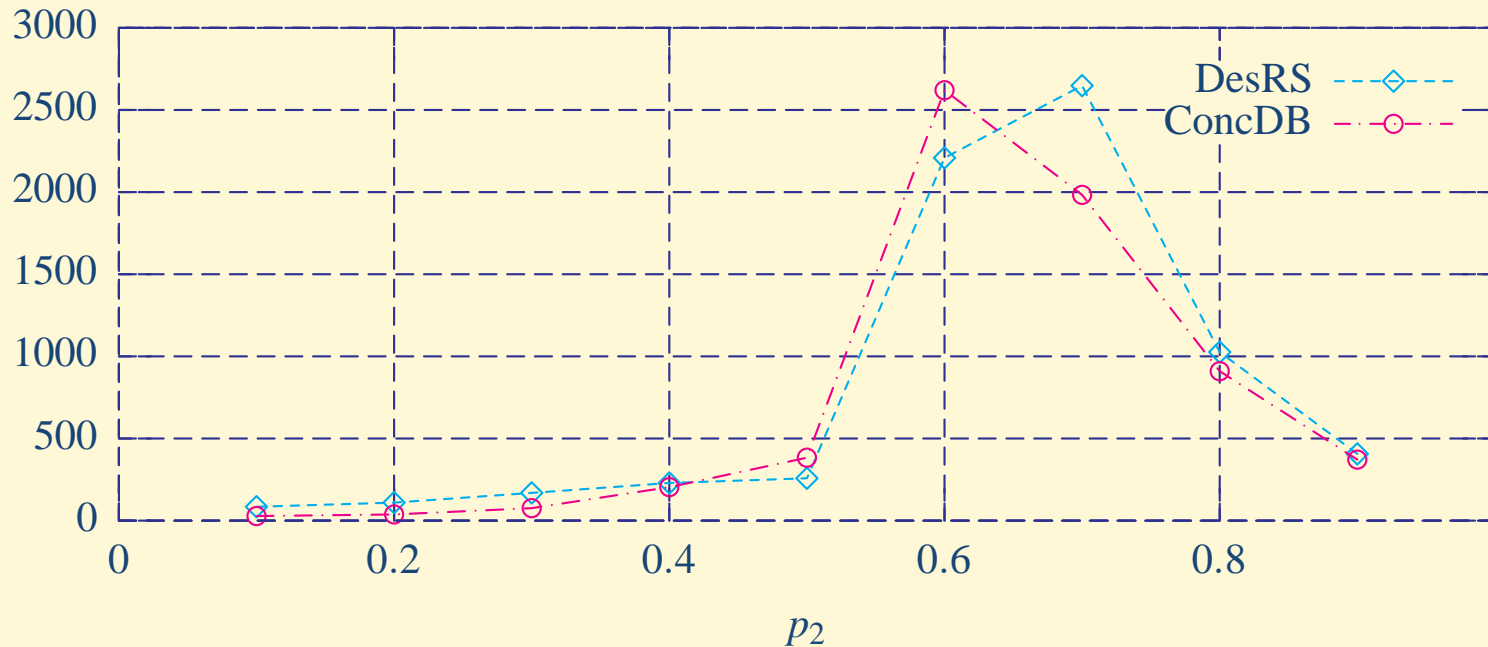
[References](#)



Experimental Results: DesRS

- DesRS, ConcDB [Zivan and Meisels, 2006] on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$

Constraint checks: DesRS and ConcDB



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

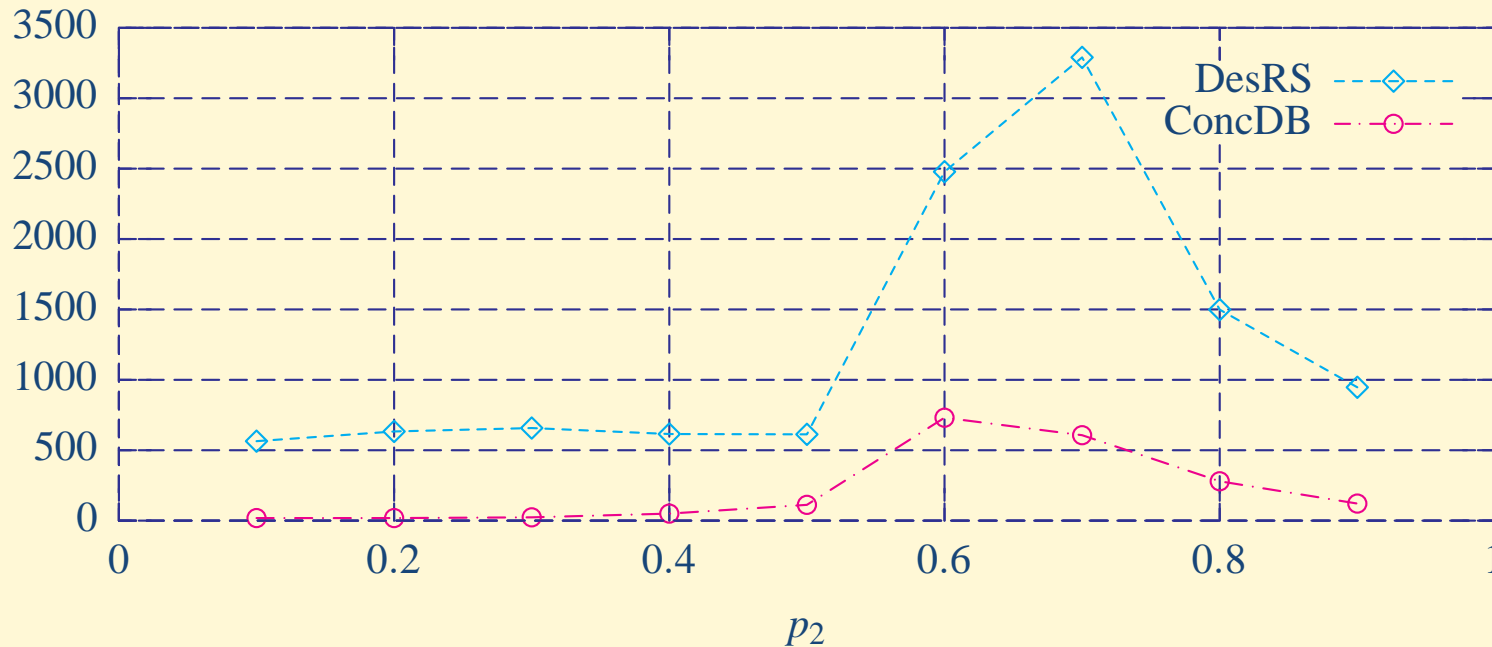
[References](#)



Experimental Results: DesRS

- DesRS, ConcDB [Zivan and Meisels, 2006] on random problems with 10 agents, domain size of 10, and $p_1 = 0.5$

Messages: DesRS and ConcDB



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

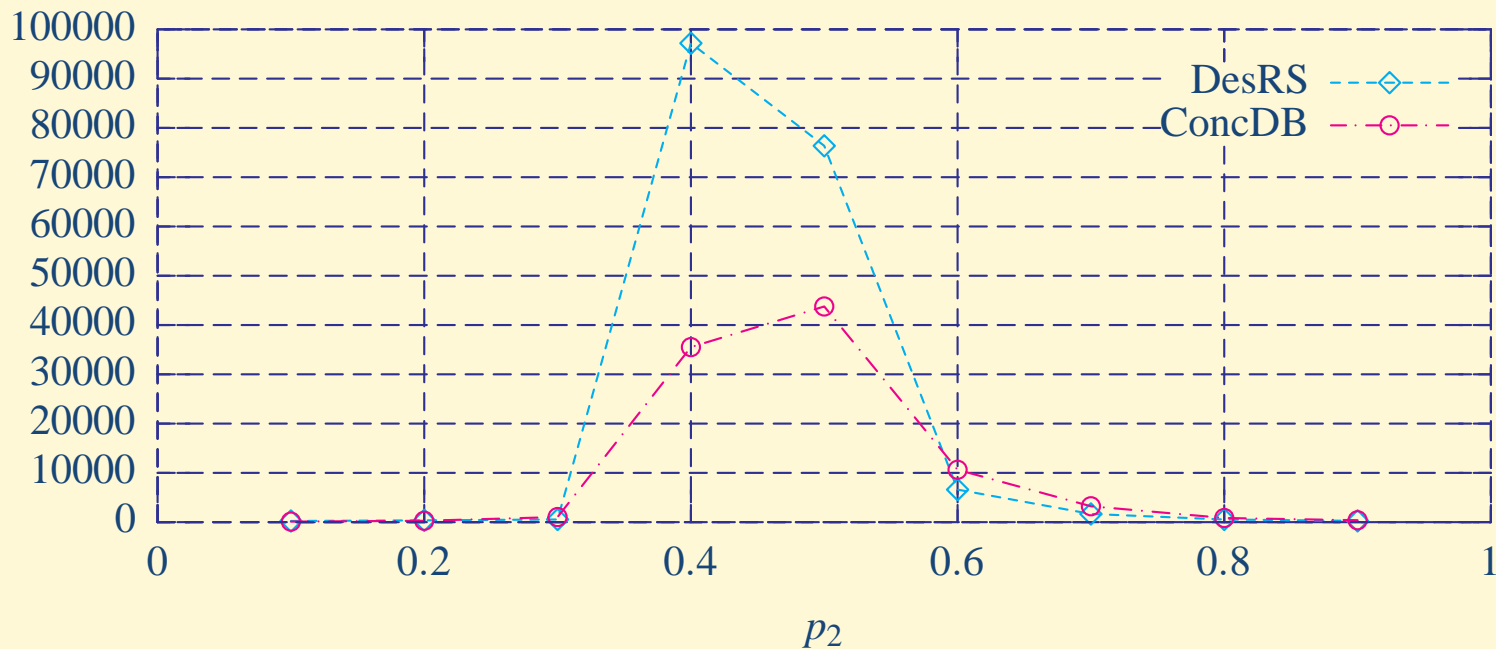
[References](#)



Experimental Results: DesRS

- DesRS, ConcDB on random problems with 20 agents, domain size of 10, and $p_1 = 0.4$

Constraint checks: DesRS and ConcDB



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

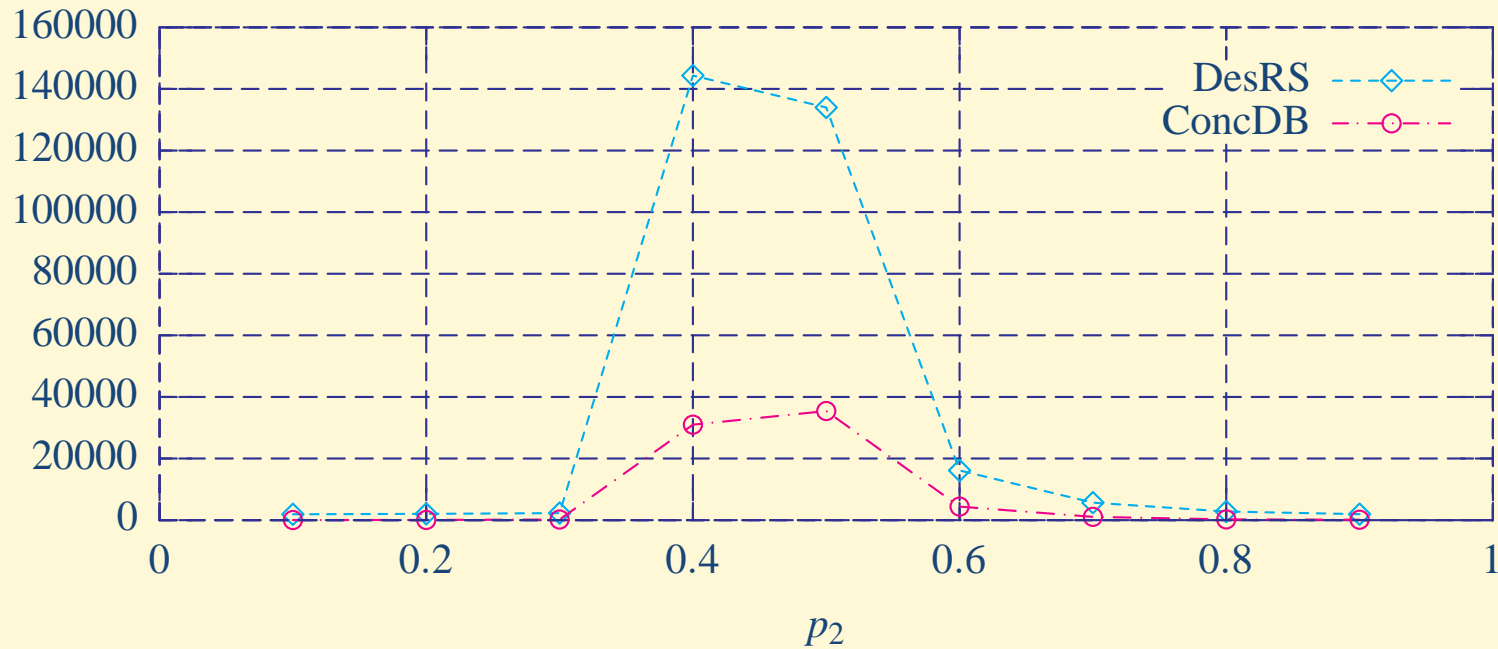
[References](#)



Experimental Results: DesRS

- DesRS, ConcDB on random problems with 20 agents, domain size of 10, and $p_1 = 0.4$

Messages: DesRS and ConcDB



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)



Importance of partition: DisHS vs. AntiDisHS

- What if we change the order of neighbors during partition (10 agents, 10 values, $p_1 = 0.5$)?

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

DisHS

DesRS, ABT

DesRS, ConcDB

AntiDisHS

AntiDesRS

[Beyond Search](#)

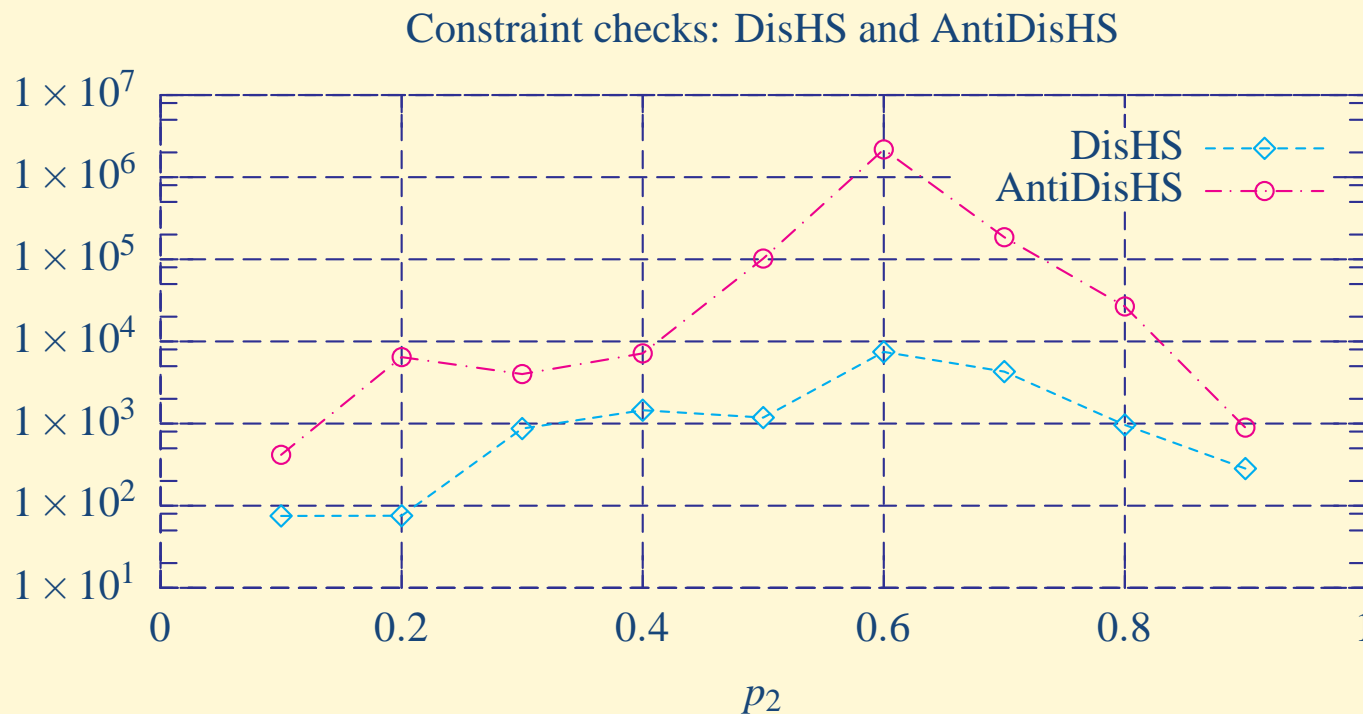
[References](#)



Experimental Results

Importance of partition: DisHS vs. AntiDisHS

- What if we change the order of neighbors during partition (10 agents, 10 values, $p_1 = 0.5$)?



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)

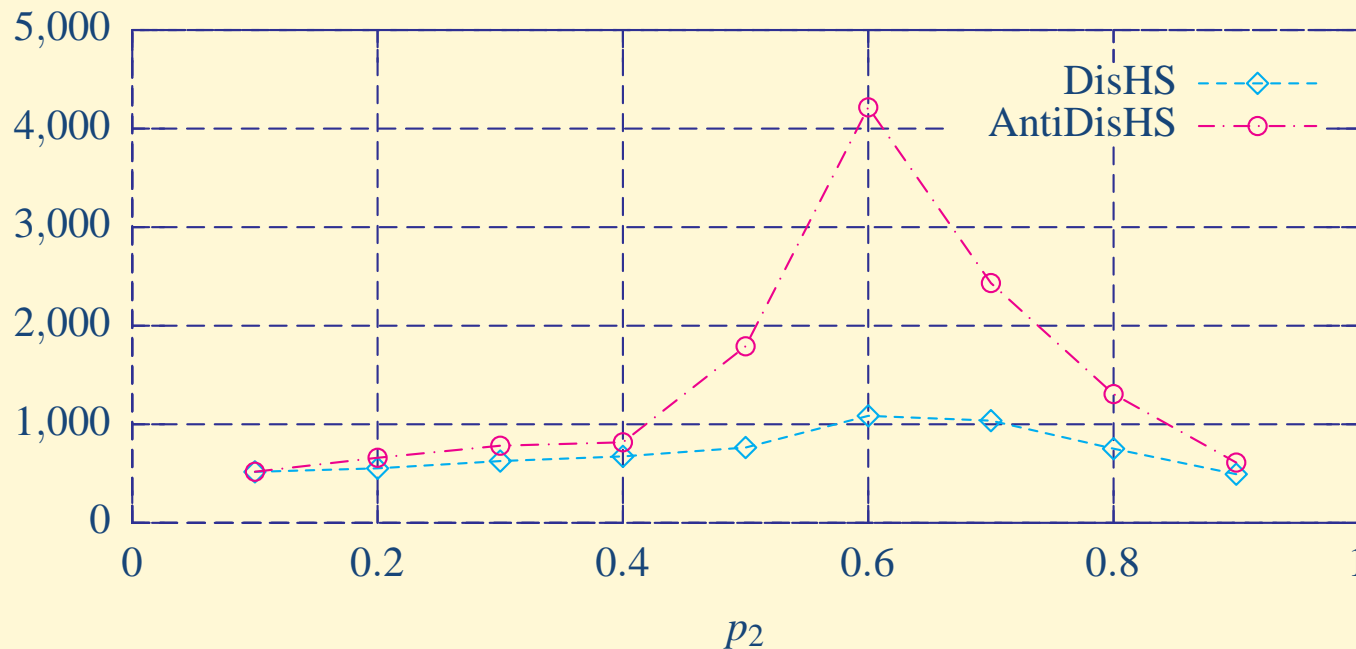


Experimental Results

Importance of partition: DisHS vs. AntiDisHS

- What if we change the order of neighbors during partition (10 agents, 10 values, $p_1 = 0.5$)?

Messages: DisHS and AntiDisHS



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)



Experimental Results

Importance of partition: DesRS vs. AntiDesRS

- What if we change the order of neighbors during partition (20 agents, 10 values, $p_1 = 0.4$)?

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)

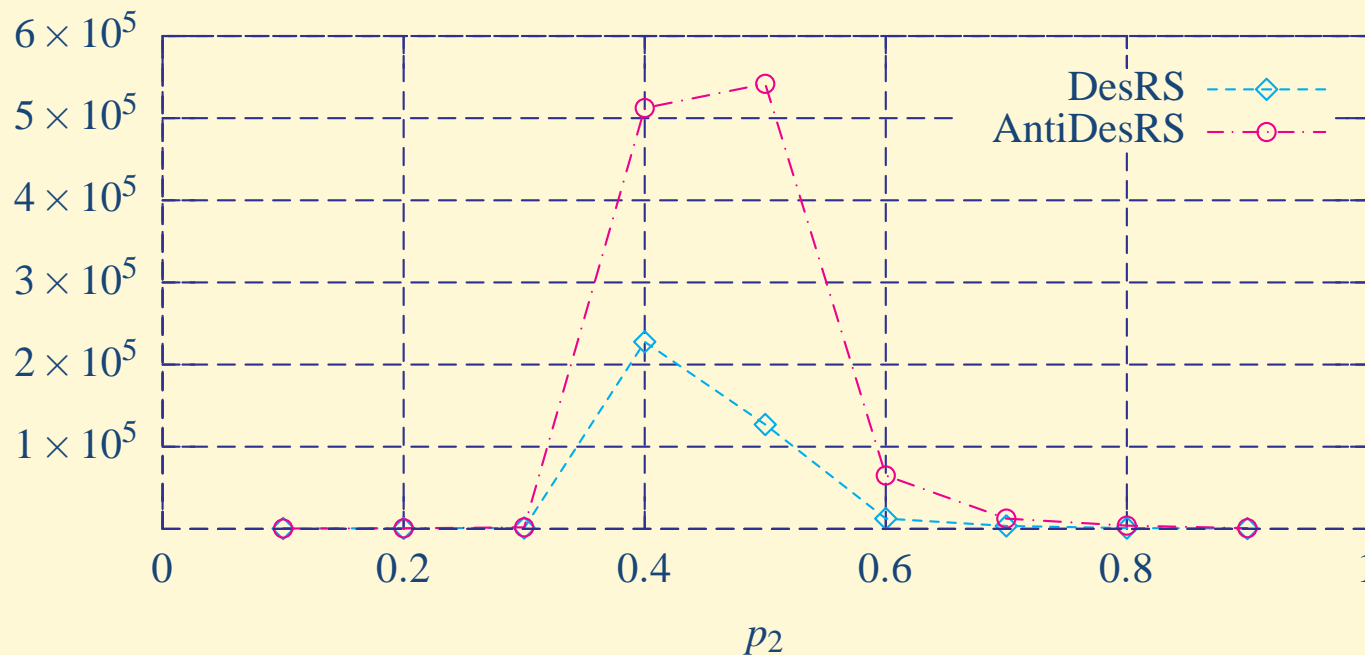


Experimental Results

Importance of partition: DesRS vs. AntiDesRS

- What if we change the order of neighbors during partition (20 agents, 10 values, $p_1 = 0.4$)?

Constraint checks: DesRS and AntiDesRS



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)

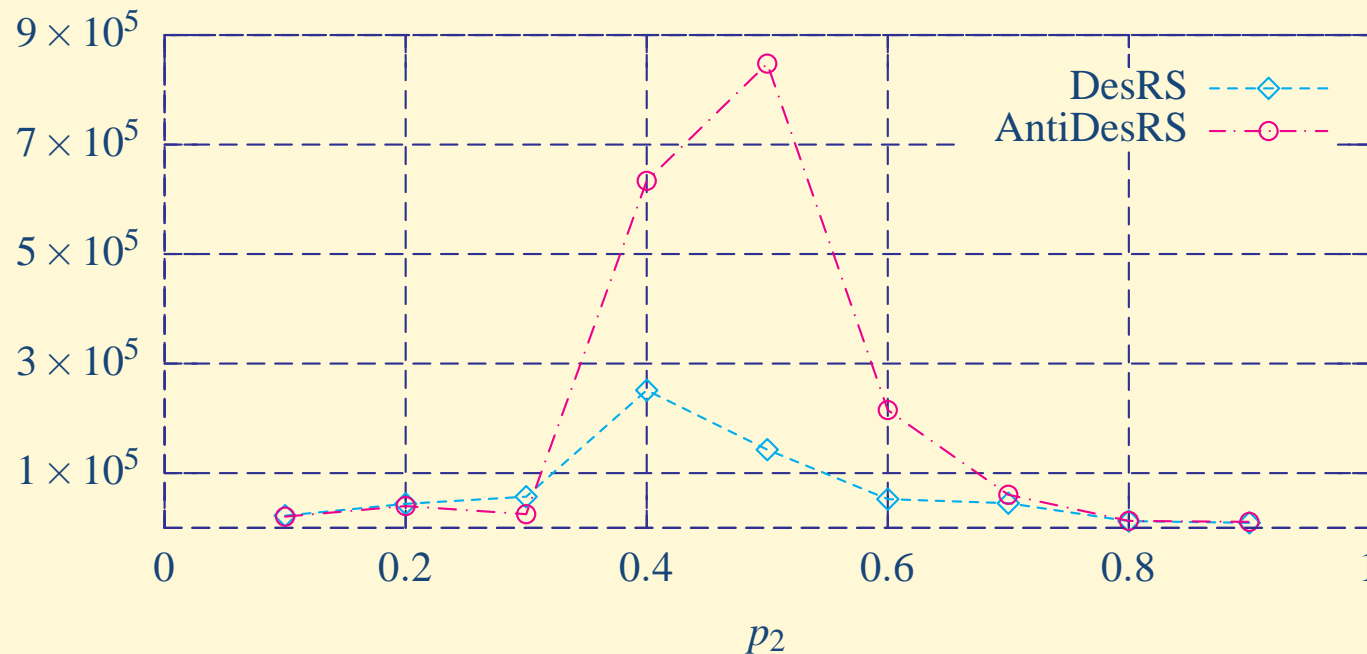


Experimental Results

Importance of partition: DesRS vs. AntiDesRS

- What if we change the order of neighbors during partition (20 agents, 10 values, $p_1 = 0.4$)?

Messages: DesRS and AntiDesRS



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[DisHS](#)

[DesRS, ABT](#)

[DesRS, ConcDB](#)

[AntiDisHS](#)

[AntiDesRS](#)

[Beyond Search](#)

[References](#)



Beyond Search

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Partition: Additional applications

- Influence in social networks

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Partition: Additional applications

- Influence in social networks
- Load balancing

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



Partition: Additional applications

- Influence in social networks
- Load balancing
- Partition is fast!

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

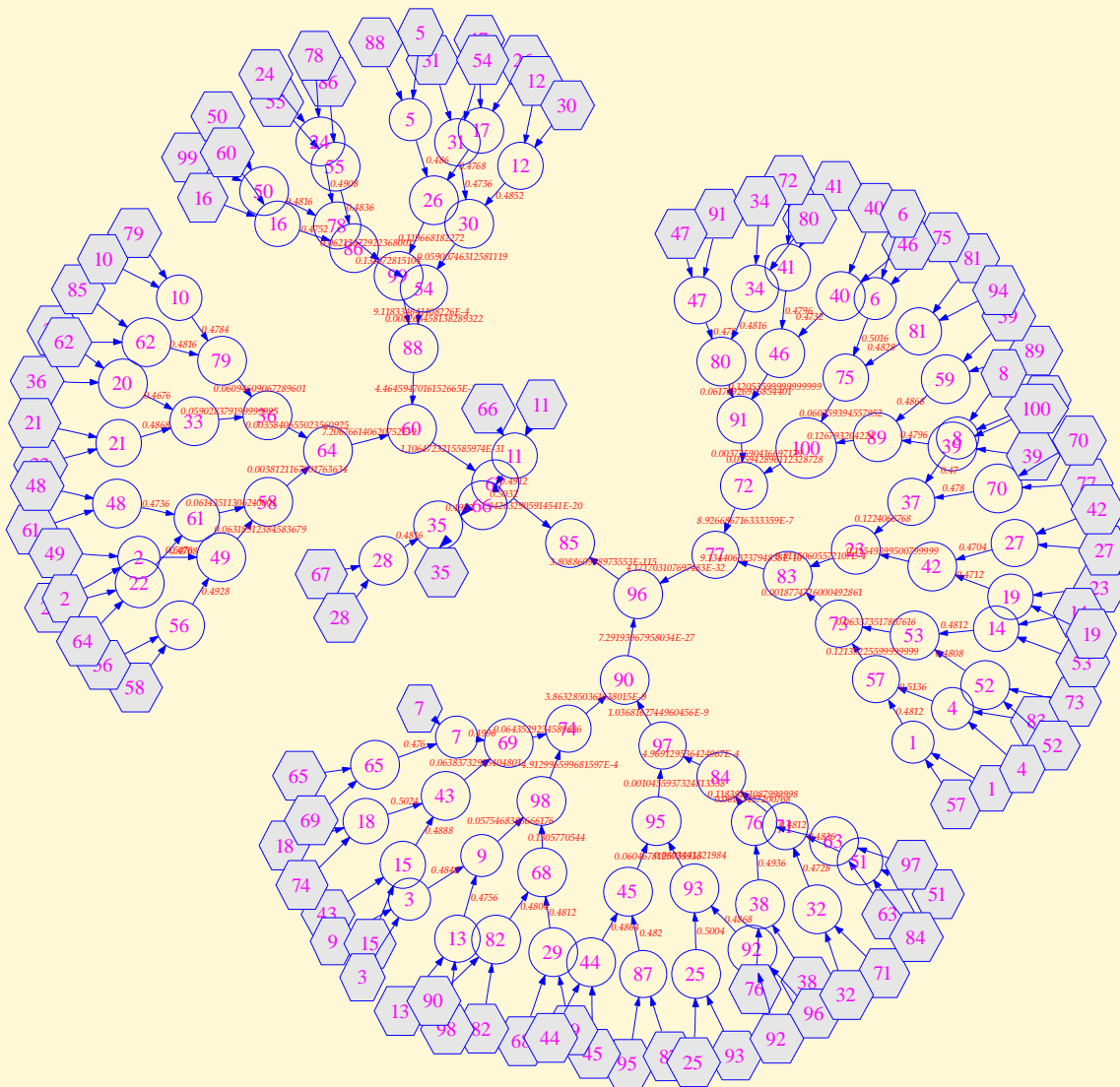
[Experimental Results](#)

[Beyond Search](#)

[References](#)



Partition: Additional applications



[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed Hierarchical Search](#)

[Descending Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)



- A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. In *Proceedings of the Third Workshop on Distributed Constraint Reasoning*, pages 86–93, Bologna, Italy, July 2002.
- M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.
- R. Zivan and A. Meisels. Concurrent search for distributed CSPs. *Artificial Intelligence Journal*, 170(4–5):440–461, Apr. 2006.

[Introduction](#)

[Motivation](#)

[Algorithms Overview](#)

[Group Partition](#)

[Distributed
Hierarchical Search](#)

[Descending
Requirements Search](#)

[Experimental Results](#)

[Beyond Search](#)

[References](#)