Automatic Construction, Maintenance, and Optimization of Dynamic Agent Organizations

Evan A. Sultanik

Advisors: William C. Regli and Ali Shokoufandeh

A Thesis Submitted to the Faculty of Drexel University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science

8 September 2010
When humans have a common goal, we naturally coalesce into an organizational structure that enables our coordination.

**Example: The Corporate Hierarchy**

- Costs/Fees
- Landmarks
- Logistics
- Hiring
- Management
- Advertising
- Logistics
- Advertising
Humans design their computer system organizations in a similar way.

Example: Server Farms

- Internet
- Firewall / IDS
- Web Server Cluster
- Database Cluster
- Load Balancer
Steiner Networks: “Nature’s Organization”

Scott Aaronson
NP-complete Problems and Physical Reality
Steiner Networks: “Nature’s Organization”

Scott Aaronson
NP-complete Problems and Physical Reality
Steiner Networks: “Nature’s Organization”

Scott Aaronson

NP-complete Problems and Physical Reality
Steiner Networks: “Nature’s Organization”

Scott Aaronson
NP-complete Problems and Physical Reality
Problem:
Given a notion of the way in which a group of software agents need to interact, what is the best organizational structure for the system that might expedite the process of coordination while maintaining a certain operating point?

- $a_1$: I need to interact with $a_2$
- $a_2$: I need to interact with $a_1$ and $a_4$
- $a_3$: I need to interact with $a_4$, $a_7$, and $a_{12}$
  ...

Steiner network
Contributions

1. Identify a set of connectivity problems—called *constrained forest problems*—that capture many multiagent organization problems.

2. Parallel/distributed algorithms for constructing constrained forests.
   - Linear (and sometimes logarithmic) messaging rounds.
   - Polynomial local runtime and memory.
   - Constant factor of optimal.

3. **Tangential Result:** For combinatorial optimization problems on finite structures, any randomly chosen feasible solution is with high probability a constant factor of optimal.

4. Examples of how to apply the framework to real-world problems (e.g., Art Gallery, Facility Location, &c.).

5. Algorithmic extensions for *dynamic* problems and problems that are more expressive than constrained forests (e.g., pseudotrees).
Outline

1. Introduction
2. The Generalized Distributed Constrained Forest Algorithm for Multidirectional Graph Search
3. Bad Things Rarely Happen to Good Graphs
4. Solving Constrained Forest Problems
5. Dynamic Agent Organizations
6. Conclusions
Distributed Multiagent Organization is hard, yet useful.

Measuring the performance of distributed algorithms is also hard.

In most cases, we want $O(n)$ communication rounds.

Approximation algorithms are often useful and sometimes necessary.

The Primal-Dual Schema is a very powerful tool for approximation.

The schema seems to be distributable...
Idea:

Represent the interaction constraints of the agents as a graph; then solve the connectivity problem using a form of search!
Search
(Foreshadowing!)

- **DFS**: Stack
- **BFS**: Queue
- **Best-first**: Priority Queue
- **A\(^*\)**: Priority Queue with Heuristic
Bidirectional Search

- Modified \textsc{goal-test} and an optimal search $\Rightarrow$ guaranteed optimality.
- Speedup from parallelism.
- \textbf{Question:} What if Erdős wants to join the party?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?
Multidirectional Graph Search?

E. A. Sultanik (Drexel University)

Dynamic Agent Organizations

2010-09-08 13 / 52
Multidirectional Graph Search

Challenges

- How do we prevent cycles?
- How do we ensure correctness/completeness?
- Optimality?

Dynamic Agent Organizations

E. A. Sultanik (Drexel University)
1: procedure MULTIDIRECTIONAL-GRAH-SEARCH(v)

Require: v is the start vertex running this search.
Ensure: H = ⟨V, E⟩ is the resulting forest.

2: \[ \tilde{V} \leftarrow \{v\} \]
3: \[ \tilde{E} \leftarrow \emptyset \]
4: \[ F \leftarrow \text{our neighbors} \quad \text{(* The fringe of our search *)} \]
5: \[ g(v) \leftarrow 0 \quad \text{for all } v \in V \quad \text{(* Initialize the path-cost function to 0 *)} \]
6: while Our interaction constraints are still unsatisfied do
7: \[ \text{Find an edge } e = \langle u, v \rangle \text{ in the fringe that minimizes } \varepsilon = w(e) - g(u) - g(v) \]
8: if u either is being or already was expanded by another search then
9: \[ \text{Merge our execution with } u\text{'s search.} \]
10: if The other search also expanded the edge \langle v, u \rangle in this round then
11: \[ \varepsilon \leftarrow \frac{\varepsilon}{2} \]
12: for all \( k \in \tilde{V} : k \text{ is incident to an edge in the fringe} \) do
13: \[ g(k) \leftarrow g(k) + \varepsilon \quad \text{(* Update the path-cost *)} \]
14: \[ F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\}) \quad \text{(* Update the fringe with } e\text{'s successors *)} \]
15: \[ \tilde{V} \leftarrow \tilde{V} \cup \{u\} \quad \text{(* Add } u\text{ to the final forest *)} \]
16: \[ \tilde{E} \leftarrow \tilde{E} \cup \{e\} \quad \text{(* Add } e\text{ to the final forest *)} \]
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAPH-SEARCH(v)

Require: v is the start vertex running this search.
Ensure: $H = (\tilde{V}, \tilde{E})$ is the resulting forest.

2: $\tilde{V} \leftarrow \{v\}$
3: $\tilde{E} \leftarrow \emptyset$
4: $F \leftarrow$ our neighbors /* The fringe of our search */
5: $g(v) \leftarrow 0$ for all $v \in V$ /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7: Find an edge $e = \langle u, v \rangle$ in the fringe that minimizes $\epsilon = w(e) - g(u) - g(v)$
8: if $u$ either is being or already was expanded by another search then
9: Merge our execution with $u$’s search.
10: if The other search also expanded the edge $\langle v, u \rangle$ in this round then
11: $\epsilon \leftarrow \frac{\epsilon}{2}$
12: for all $k \in \tilde{V}$ : $k$ is incident to an edge in the fringe do
13: $g(k) \leftarrow g(k) + \epsilon$ /* Update the path-cost */
14: $F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\})$ /* Update the fringe with $e$’s successors */
15: $\tilde{V} \leftarrow \tilde{V} \cup \{u\}$ /* Add $u$ to the final forest */
16: $\tilde{E} \leftarrow \tilde{E} \cup \{e\}$ /* Add $e$ to the final forest */
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAFH-SEARCH (v)
   Require: v is the start vertex
   Ensure: H = (\tilde{V}, \tilde{E}) is the resulting forest.
2: \tilde{V} ← \{v\}
3: \tilde{E} ← \emptyset
4: F ← our neighbors  /* The fringe of our search */
5: g(v) ← 0 for all v ∈ V  /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7:     Find an edge e = \langle u, v \rangle in the fringe that minimizes \epsilon = w(e) − g(u) − g(v)
8:     if u either is being or already was expanded by another search then
9:         Merge our execution with u’s search.
10:    if The other search also expanded the edge \langle v, u \rangle in this round then
11:       \epsilon ← \frac{\epsilon}{2}
12:     for all k ∈ \tilde{V} : k is incident to an edge in the fringe do
13:         g(k) ← g(k) + \epsilon  /* Update the path-cost */
14:     F ← (F \{e\}) ∪ δ({u})  /* Update the fringe with e’s successors */
15:     \tilde{V} ← V ∪ \{u\}  /* Add u to the final forest */
16:     \tilde{E} ← \tilde{E} ∪ \{e\}  /* Add e to the final forest */
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAFP-SEARCH(v)

Require: v is the start vertex running this search.

Ensure: H = ⟨Ṽ, Ẽ⟩ is the resulting forest.

2: Ṽ ← {v}
3: Ẽ ← ∅
4: F ← our neighbors /* The fringe of our search */
5: g(v) ← 0 for all v ∈ V /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7: Find an edge e = ⟨u, v⟩ in the fringe that minimizes ε = w(e) − g(u) − g(v)
8: if u either is being or already was expanded by another search then
9: Merge our execution with u’s search.
10: if The other search also expanded the edge ⟨v, u⟩ in this round then
11: ε ← ε / 2
12: for all k ∈ Ṽ : k is incident to an edge in the fringe do
13: g(k) ← g(k) + ε /* Update the path-cost */
14: F ← (F \ {e}) ∪ δ({u}) /* Update the fringe with e’s successors */
15: Ṽ ← Ṽ ∪ {u} /* Add u to the final forest */
16: Ẽ ← Ẽ ∪ {e} /* Add e to the final forest */
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAPH-SEARCH(v)

Require: v is the start vertex running this search.
Ensure: H = (V, E) is the resulting forest.

2: \( \tilde{V} \leftarrow \{v\} \)
3: \( \tilde{E} \leftarrow \emptyset \)
4: \( F \leftarrow \) our neighbors
5: \( g(v) \leftarrow 0 \) for all \( v \in V \)
6: while Our interaction constraints are still unsatisfied do
7: \( \text{Find an edge } e = \langle u, v \rangle \text{ in the fringe that minimizes } \epsilon = w(e) - g(u) - g(v) \)
8: if u either is being or already was expanded by another search then
9: Merge our execution with u’s search.
10: if The other search also expanded the edge \( \langle v, u \rangle \) in this round then
11: \( \epsilon \leftarrow \frac{\epsilon}{2} \)
12: for all \( k \in \tilde{V} : k \) is incident to an edge in the fringe do
13: \( g(k) \leftarrow g(k) + \epsilon \) /* Update the path-cost */
14: \( F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\}) \) /* Update the fringe with e’s successors */
15: \( \tilde{V} \leftarrow \tilde{V} \cup \{u\} \) /* Add u to the final forest */
16: \( \tilde{E} \leftarrow \tilde{E} \cup \{e\} \) /* Add e to the final forest */
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAPH-SEARCH(v)

Require: v is the start vertex running this search.
Ensure: \( H = \langle \tilde{V}, \tilde{E} \rangle \) is the resulting forest.

2: \( \tilde{V} \leftarrow \{v\} \)
3: \( \tilde{E} \leftarrow \emptyset \)
4: \( F \leftarrow \) our neighbors /* The fringe of our search */
5: \( g(v) \leftarrow 0 \) for all \( v \in V \) /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7: Find an edge \( e = \langle u, v \rangle \) in the fringe that minimizes \( \varepsilon = w(e) - g(u) - g(v) \)
8: if u either is being or already was expanded by another search then
9: Merge our execution with u’s search.
10: if The other search also expanded the edge \( \langle v, u \rangle \) in this round then
11: \( \varepsilon \leftarrow \frac{\varepsilon}{2} \)
12: for all \( k \in \tilde{V} : k \) is incident to an edge in the fringe do
13: \( g(k) \leftarrow g(k) + \varepsilon \) /* Update the path-cost */
14: \( F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\}) \) /* Update the fringe with e’s successors */
15: \( \tilde{V} \leftarrow \tilde{V} \cup \{u\} \) /* Add u to the final forest */
16: \( \tilde{E} \leftarrow \tilde{E} \cup \{e\} \) /* Add e to the final forest */
Generalized Distributed Constrained Forest Algorithm
for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAPH-SEARCH(v)

Require: $v$ is the start vertex running this search.

Ensure: $H = \langle \tilde{V}, \tilde{E} \rangle$ is the resulting forest.

2: $\tilde{V} \leftarrow \{v\}$
3: $\tilde{E} \leftarrow \emptyset$
4: $F \leftarrow$ our neighbors  / * The fringe of our search */
5: $g(v) \leftarrow 0$ for all $v \in V$  / * Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7:     Find an edge $e = \langle u, v \rangle$ in the fringe that minimizes $\varepsilon = w(e) - g(u) - g(v)$
8:     if $u$ either is being or already was expanded by another search then
9:         Merge our execution with $u$'s search.
10:    if The other search also expanded the edge $\langle v, u \rangle$ in this round then
11:        $\varepsilon \leftarrow \frac{\varepsilon}{2}$
12:    for all $k \in \tilde{V}$ : $k$ is incident to an edge in the fringe do
13:        $g(k) \leftarrow g(k) + \varepsilon$  / * Update the path-cost */
14:    $F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\})$  / * Update the fringe with $e$'s successors */
15:    $\tilde{V} \leftarrow \tilde{V} \cup \{u\}$  / * Add $u$ to the final forest */
16:    $\tilde{E} \leftarrow \tilde{E} \cup \{e\}$  / * Add $e$ to the final forest */

Gradient!

This is the potential function that will ensure 2-OPT.
Generalized Distributed Constrained Forest Algorithm

Dual Variables

The path-cost implicitly initializes the dual variables.

1: procedure MULTIDIRECTIONAL-GRAPH-SEARCH(v)
Require: v is the start vertex running this search.
Ensure: H = (\tilde{V}, \tilde{E}) is the resulting forest.
2: \tilde{V} ← {v}
3: \tilde{E} ← ∅
4: F ← our neighbors  /* The fringe of our search */
5: g(v) ← 0 for all v ∈ V  /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7: Find an edge e = ⟨u, v⟩ in the fringe that minimizes ε = w(e) − g(u) − g(v)
8: if u either is being or already was expanded by another search then
9: Merge our execution with u’s search.
10: if The other search also expanded the edge ⟨v, u⟩ in this round then
11: ε ← \frac{ε}{2}
12: for all k ∈ \tilde{V} : k is incident to an edge in the fringe do
13: g(k) ← g(k) + ε  /* Update the path-cost */
14: F ← (F \ {e}) ∪ δ({u})  /* Update the fringe with e’s successors */
15: \tilde{V} ← V ∪ {u}  /* Add u to the final forest */
16: \tilde{E} ← \tilde{E} ∪ {e}  /* Add e to the final forest */
Generalized Distributed Constrained Forest Algorithm for Multidirectional Graph Search

1: procedure MULTIDIRECTIONAL-GRAH-SEARCH(v)

Require: \( v \) is the start vertex running this search.
Ensure: \( H = \langle \tilde{V}, \tilde{E} \rangle \) is the resulting forest.

2: \( \tilde{V} \leftarrow \{v\} \)
3: \( \tilde{E} \leftarrow \emptyset \)
4: \( F \leftarrow \) our neighbors /* The fringe of our search */
5: \( g(v) \leftarrow 0 \) for all \( v \in V \) /* Initialize the path-cost function to 0 */
6: while Our interaction constraints are still unsatisfied do
7: Find an edge \( e = \langle u, v \rangle \) in the fringe that minimizes \( \varepsilon = w(e) - g(u) - g(v) \)
8: if \( u \) either is being or already was expanded by another search then
9: Merge our execution with \( u \)’s search.
10: if The other search also expanded the edge \( \langle v, u \rangle \) in this round then
11: \( \varepsilon \leftarrow \frac{\varepsilon}{2} \)
12: for all \( k \in \tilde{V} : k \) is incident to an edge in the fringe do
13: \( g(k) \leftarrow g(k) + \varepsilon \) /* Update the path-cost */
14: \( F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\}) \) /* Update the fringe with \( e \)’s successors */
15: \( \tilde{V} \leftarrow \tilde{V} \cup \{u\} \) /* Add \( u \) to the final forest */
16: \( \tilde{E} \leftarrow \tilde{E} \cup \{e\} \) /* Add \( e \) to the final forest */

Pushing Up the Duals
Implicitly sets \( y_{\tilde{V}} \leftarrow y_{\tilde{V}} + \varepsilon \).

Department of Computer Science
The memory usage at a node is bounded above by $O(|E|) = O\left(\frac{n^2-n}{2}\right)$.

The local computation is bounded above by $O(|V| \times |E|) = O\left(\frac{n^3-n^2}{2}\right)$.

Selecting nodes in parallel will not induce a cycle in the constrained forest. [Proof Sketch]

The algorithm is correct and complete. [Proof Sketch]

If there exists an $\tilde{\omega}$ such that all edge weights are in the range $[\tilde{\omega}, \frac{3}{2}\tilde{\omega}]$ then the cost of the final solution is 2-optimal. [Proof Sketch]

If the number of edges that will later be added to an active component is bounded above by 2, then the solution will also be 2-optimal. [Proof Sketch]
## Results

(Bounds on the Total Number of Rounds)

<table>
<thead>
<tr>
<th>All component unions are mutual and $\alpha \geq \frac{n}{2}$</th>
<th>Otherwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>All agents can communicate with each other in $O(1)$ rounds</td>
<td>$O(\log n)$ $\dagger$</td>
</tr>
<tr>
<td>Routing is required for communication</td>
<td>$O(\log^2 n)$ $\ddagger$</td>
</tr>
</tbody>
</table>

$\dagger$ This achieves the lower bound on the time-approximation tradeoff [Elkin, 2004].

$\ddagger$ Convergence to these bounds is sublinear.
Time-Approximation Tradeoff

- Let $\omega_{\text{max}}$ be the ratio of the heaviest/lightest edges in the graph.
- If we know an upper bound on $\omega_{\text{max}}$, it turns out we can find an $\varepsilon$-optimal solution at the expense of additional communication/computation rounds.
- For example, ensuring a 2-optimal solution requires only
  \[
  O \left( \log \left( \frac{\log \left( \frac{2}{3} \omega_{\text{max}} \right)}{\log \log n} \right) (n) \right)
  \]
  additional rounds.
- If $\exists c : \omega_{\text{max}} \leq \frac{3}{2} \log^c(n)$, we only need $O(\log^c n)$ additional rounds.
- Fully polynomial-time approximation.
Consider a combinatorial optimization problem on a finite structure, such as this random lattice:
Weight the edges ...
Weight the edges . . . and sort them:
If an algorithm picks at most $\ell$ edges, then the cost of the algorithm’s solution is bounded above by the sum of the $\ell$ heaviest edges.
If the optimal solution contains at least $m$ edges, then the cost of the optimal solution is bounded below by the sum of the $m$ lightest edges.
\[ \ell \sum_{i=0}^{\ell} w_i \bigg/ \sum_{j=0}^{m} w_j \] is an upper bound on the constant of approximation!
Vector random variable for the edge weights: \( \mathbf{X} = [X_1, X_2, \ldots, X_n]^T \).

Let \( Z \) be the ratio distribution of the trimmed sums of \( \mathbf{X} \)'s order statistics:

\[
Z = \frac{\sum_{i=1}^{\ell} X_i}{\sum_{j=1}^{m} X_j}.
\]

The expected value of \( Z \) is therefore an upper bound on the expected value of the constant of approximation. *Quod erat requisitum.*

This is an open problem!
A Recurrence Relation

However, it is possible to devise a recurrence which we can evaluate using numerical methods:

\[
P \left[ \sum_{i=1}^{m} X_{(n,i)} = x \right] =
\begin{cases}
P \left[ X_{(n,1)} = x \right] & \text{if } m = 1, \\
\int_{0}^{x} P \left[ X_{(n,m)} = s \right] P \left[ \sum_{i=1}^{m-1} X_{(n,i)} = x - s \right] \, ds & \text{otherwise.}
\end{cases}
\]
Example: The Standard Uniform Distribution

\[ P \left( \sum_{i=n-\ell+1}^{n} U(0,1)(a_i) \leq x \right) \]

Figure: CDF for the distribution of the sum of the \( \ell \) largest order statistics of a sample of size 10 from the standard uniform distribution.
Results

- For continuous uniform distributions the quotient of the expected values of the nominator and denominator in $Z$ should give a relatively unbiased estimation of the true expected value of $Z$, which itself should have very low variance.

- For the uniform, normal, and exponential distributions:

  \[
  \mathbb{E}[Z] = \frac{\ell \mathbb{E}[X]}{m \mathbb{E}[X]} = \frac{\ell}{m}.
  \]

- For large $\ell$ and $m$, the same holds in general.
Implications of $E[Z] = \ell \div m$

- Surprise!
- If we know the size of the optimal solution is bounded below by $m$, then any randomly chosen solution of size at most $\ell$ will, on average, be $\frac{\ell}{m}$ times optimal.

Example

The Steiner Network Problem

- If there are $\alpha$ terminals, then we know that the optimal solution must have at least $\left\lfloor \frac{\alpha}{2} \right\rfloor$ edges.
- Any feasible solution to the problem is going to be an acyclic graph, which will have at most $n - 1$ edges.
- Therefore, any randomly chosen feasible solution to the Steiner network problem will be, on average, $\frac{2n-2}{\alpha}$ times optimal.
- If $\alpha = n$, feasible solutions are with high probability 2-Optimal.
Implications of $E[Z] = \ell \div m$

- Surprise!
- If we know the size of the optimal solution is bounded below by $m$, then any randomly chosen solution of size at most $\ell$ will, on average, be $\frac{\ell}{m}$ times optimal.

Example

The Minimum Spanning Tree Problem

- Assuming the graph is connected, the optimal solution has exactly $n - 1$ edges.
- Any feasible solution will have exactly $n - 1$ edges.
- Therefore, any randomly chosen feasible solution to the minimum spanning tree problem will with high probability be optimal.
Experimental Setup

1. Generated 608 random 200-vertex graphs of varying edge density using the Erdős-Rényi model.

2. The algorithm was then run with best case concurrency (i.e., one agent per vertex).

Results

![Graph showing the relationship between normalized cost and edge density. The x-axis represents edge density ranging from 0 to 1, and the y-axis represents normalized cost ranging from 0 to 0.14. The graph shows a decreasing trend as edge density increases.]
Experimental Setup

1. Fixed the edge density at 0.1 (this is the worst case).
2. Re-ran the experiment with a varying number of terminals.

Results

 GetMessageing Rounds vs. % Vertices that are Terminals
Experimental Setup

1. Fixed the edge density at 0.1 (this is the worst case).
2. Re-ran the experiment with a varying number of terminals.

Results

Approximation Bound vs. % Vertices that are Terminals

The graph shows the relationship between the approximation bound and the percentage of vertices that are terminals.
1: \textbf{procedure} \textsc{Multidirectional-Graph-Search}(v) \\
\textbf{Require:} v is the start vertex running this search. \\
\textbf{Ensure:} \( H = \langle \tilde{V}, \tilde{E} \rangle \) is the resulting forest.

2: \( \tilde{V} \leftarrow \{v\} \)
3: \( \tilde{E} \leftarrow \emptyset \)
4: \( F \leftarrow \) our neighbors \hspace{1em} /* The fringe of our search */
5: \( g(v) \leftarrow 0 \) for all \( v \in V \) \hspace{1em} /* Initialize the path-cost function to 0 */
6: \textbf{while} Our interaction constraints are still unsatisfied \textbf{do}
7: \hspace{1em} Find an edge \( e = \langle u, v \rangle \) in the fringe that minimizes \( \varepsilon = w(e) - g(u) - g(v) \)
8: \hspace{1em} \textbf{if} u either is being or already was expanded by another search \textbf{then}
9: \hspace{2em} Merge our execution with u’s search.
10: \hspace{1em} \textbf{if} The other search also expanded the edge \( \langle v, u \rangle \) in this round \textbf{then}
11: \hspace{2em} \( \varepsilon \leftarrow \frac{\varepsilon}{2} \)
12: \hspace{1em} \textbf{for all} \( k \in \tilde{V} : k \) is incident to an edge in the fringe \textbf{do}
13: \hspace{2em} \( g(k) \leftarrow g(k) + \varepsilon \) \hspace{1em} /* Update the path-cost */
14: \hspace{1em} \( F \leftarrow (F \setminus \{e\}) \cup \delta(\{u\}) \) \hspace{1em} /* Update the fringe with e’s successors */
15: \hspace{1em} \( \tilde{V} \leftarrow \tilde{V} \cup \{u\} \) \hspace{1em} /* Add u to the final forest */
16: \hspace{1em} \( \tilde{E} \leftarrow \tilde{E} \cup \{e\} \) \hspace{1em} /* Add e to the final forest */

Goal-Test Function

If the goal-test function is a \textit{proper function}, we can solve many other problems!
## Example Constrained Forest Problems

\[
\begin{align*}
\min & \quad \sum_{e \in E} w(e)x_e \\
\text{s.t.} & \quad \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \\
& \quad x_e \geq 0, \quad \forall e \in E,
\end{align*}
\]

\[
\begin{align*}
\max & \quad \sum_{S \subset V} f(S)y_S \\
\text{s.t.} & \quad \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \\
& \quad y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset,
\end{align*}
\]

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect matching</td>
<td>Find a minimum-cost set of non-adjacent edges that cover all vertices. $</td>
</tr>
<tr>
<td>$T$-join</td>
<td>Given an even subset $T$ of vertices, find a minimum-cost set of edges that has odd degree at vertices in $T$ and even degree at vertices not in $T$. $</td>
</tr>
<tr>
<td>Minimum spanning tree/forest</td>
<td>Find a minimum weight forest that maximizes connectivity between vertices. $\exists u \in S, v \notin S : u \sim v \in G$</td>
</tr>
<tr>
<td>Generalized Steiner tree</td>
<td>Find a minimum-cost forest that connects all vertices in $T_i$ for $i = 1, \ldots, p$. $\exists i \in {1, \ldots, p} : \emptyset \neq S \cap T_i \neq T_i$.</td>
</tr>
<tr>
<td>Point-to-point connection</td>
<td>Given a set $C = {c_1, \ldots, c_p}$ of sources and a set $D = {d_1, \ldots, d_p}$ of destinations in a graph $G = (V, E)$, find a minimum-cost set $F$ of edges such that each source-destination pair is connected in $F$. $</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Find a minimum-cost collection of vertex-disjoint trees, paths, or cycles that cover all vertices. $S \not\equiv 0 \pmod{k}$.</td>
</tr>
<tr>
<td>Location design/routing</td>
<td>Select depots among a subset $D$ of vertices of a graph $G = (V, E)$ and cover all vertices in $V$ with a set of cycles, each containing a selected depot, while minimizing the sum of the fixed costs of opening depots and the sum of the costs of the edges in the cycles. $\emptyset \neq S \subseteq V$.</td>
</tr>
</tbody>
</table>
Example Domains

Corporations
- Cost/Fees
- Logistics
- Hiring
- Advertising
- Management

Robotic Teaming
- Object Tracking
- Path Planning
- Team Coordination

Sensor Networks
- Target Tracking
- Network Topology
- Communication Protocols

Service Composition
- Agent 1
  - {R, G, B}
- Agent 2
  - {R, G, B}
- Agent 3
  - {R, G, B}
- Agent 4
  - {R, G, B}

Dist. Constraint Reasoning
- Constraint Checking
- Resource Allocation
- Task Scheduling
Group of mobile robots each equipped with a wireless access point.

Objective of the robots: maximally cover an area with the wireless network.

In order to save power: Choose a maximum subset of robots that can lower their transmit power while still retaining coverage.

E. Sultanik, A. Shokoufandeh, and W. Regli
Art Gallery Problems

Example

Find the minimum number of guards required to observe the interior of a polygonal area.

Variants

- Guards in the interior.
- Treasures.
- Non-uniform cost for stationing a guard.
- NP-COMPLETE.
Augment each vertex with a special guard vertex (“●”).
Equivalence as a Connectivity Problem

Weight the new edges with the cost of guarding from that vertex.
Weight the original visibility graph edges 0.

Connectivity Problem:

1. Every node is connected to a node by a path of length \( \leq 2 \);
2. The subgraph's weight is minimized.
Connectivity Problem: Find an acyclic subgraph such that:
1. every ● is connected to a ● by a path of length ≤ 2; and
2. the subgraph’s weight is minimized.
**Round 0: All Vertices are Unguarded ("o")**

The diagram shows a network of vertices connected by lines. In Round 0, all vertices are unguarded.
Round 1: Unguarded components add cut-edge of min. potential.
Round 2: Unguarded components add cut-edge of min. potential.
Round 3: Unguarded components add cut-edge of min. potential.
**Round 4:** Unguarded components add cut-edge of min. potential.
Round 5: Unguarded components add cut-edge of min. potential.
Round 6: All nodes are guarded, so we terminate.
**Round 6:** All nodes are guarded, so we terminate.
Empirical Analysis
(Art Gallery)

Y-axis values represent the constant of approximation; lower values are better, with 1.0 being the optimal solution.

Boxes represent the second and third quartiles of each distribution.

Overall mean constant of approximation: $3.13 \pm 0.36$.

High probability that the algorithm will produce a solution with a constant approximation bound regardless of the problem size.
**Ultimate Goal**

Multiagent organization algorithms that can handle *dynamically changing* problems.

The distributed counterpart to *online algorithms* in the sequential model.

**Approach**

Provide algorithmic extensions that allow for addition/update/removal of vertices/edges via *superstabilization* and *backtracking*.
In certain well-defined cases, after a change to a problem the algorithm can be continued and is still guaranteed to produce a 2-optimal solution.

- If the weights of all of $v$’s incident edges are greater than or equal to the slack of all of their neighboring vertices’ fringe nodes:

In all other cases, we can backtrack to the most recent round during which the conditions allowed for the dynamic modification.

- Worst case: backtrack to the start, which is only $O(n)$ rounds.
- Backtracking only increases memory/computation polynomially.
Pseudotree Construction

**Example:** the Pseudotree Construction Problem

Minimum depth rooted tree such that every pair of neighbors in the interaction graph is either an ancestor or descendant of one another in the tree.

Important for Distributed Constraint Reasoning.

Not representable using proper functions because

1. pseudotrees are rooted; and
2. a global invariant must be maintained over the ancestor/descendant relationships in the tree.

We give another algorithmic extension, called Mobed, that addresses these problems.

---

R. Lass, E. Sultanik, & W. Regli
Dynamic Distributed Constraint Reasoning.
Properties of Mobed

Methods for:
- Agent removal;
- Hierarchy merger; and
- Initial hierarchy generation.

Advantages to the Approach over Alternatives (e.g., DFS)
- Same asymptotic runtime as DFS: $O(n)$ (for a single addition).
- Runtime is constant if # of interaction graph neighbors is $O(1)$.
- Edit distance is bounded at 2 (for unit cost operations).

Experimental Goals
1. Validate $O(n)$ theoretical runtime of Mobed; and
2. Compare solution stability between Mobed and DFS.
Summary of the Mobed Experimental Results

- Linear communication rounds is confirmed.
- No significant difference between hierarchy width.
- Up to 300% improvement over DFS in edit distance for Erdős-Rényi graphs.
- Almost completely dominant for Barabási-Albert small world graphs.

See the data.

E. Sultanik, R. Lass, & W. Regli
Dynamic Configuration of Agent Organizations.
A large family of multiagent organization problems—collectively called \textit{constrained forest problems}—can be solved efficiently in a distributed manner.

Generalized distributed constrained multidirectional graph search algorithm based on the primal-dual schema.

Distributed algorithms can approximate a solution in linear rounds and, under certain well defined conditions, can even quiesce in polylogarithmic or even logarithmic rounds.

Parallel algorithm in the CREW PRAM model.

Many of these problems are \textbf{NP-HARD} and \textbf{P-COMPLETE}.

Can ensure $\epsilon$-optimality at the expense of additional runtime.

Fully polynomial-time approximation.
Even if the conditions of the theoretical guarantees of 2-optimality are not met, the solutions produced by the algorithm are with high probability 2-optimal.

In fact, any randomly chosen feasible solution to a combinatorial optimization problem on a finite structure is likely to be a constant factor of optimal.

**Example**: Any randomly chosen feasible solution to the minimum spanning tree problem is with high probability optimal.
Examples of how the framework can be instantiated for various real-world problems.

Empirical results for average case performance.

Steiner Network, Art Gallery, Location Design, &c.

The decision versions of many art gallery variants are APX-HARD, however, whether they are in APX is an open problem. By identifying a class of art gallery problems that are amenable to 2-approximation, we have discovered that this class of problems is also in APX (and is thereby APX-COMPLETE).
Dynamic extensions to the algorithm: Vertex/Edge...
- addition;
- removal; and
- re-weighting.

Superstabilizing.

Mobed: an extension for problems that cannot be captured using proper functions.

Mobed exhibited up to a 300% performance improvement over its rival: distributed DFS.
Future Work

- Much work to be done in studying hierarchy generation techniques that better balance the tradeoff between computational efficiency/messaging, edit distance, and privacy, and also methods to maintain other invariants on the hierarchy’s topology.
- Generalizing the algorithms for more expressive well behaved functions (e.g., well spaced, supermodular, submodular, and integer valued functions).
- Probabilistic analysis of the length of the chain of backtracking required for incremental updates if the sufficient conditions are not met.
- Open question: Generalization of the quality of the solution for primal-dual methods as applied to constrained forest problems?
- Algorithms of this ilk are a fruitful method for achieving speedups in constrained search and many other areas of multiagent systems through parallelism.
Given the organizational structure employed by a set of agents, what is the underlying set of constraints that govern their interaction? (In a sense, this is the inverse of the problem explored in this dissertation.)

Given a *predictive* model of the communication constraints between agents, to what extent is it possible to construct an agent organization or social hierarchy that better enables coordination?

To what extent do the existing techniques for social modeling apply in cooperative, non-adversarial systems?

If we allow for adversarial agents, how can these techniques be extended to accommodate varying levels of trust?
In some organizational topologies, certain agents will have more "power" or "social influence" than others. Can one discuss (or quantize) the social influence between agents?

Given a communications protocol, is it possible to place an *a priori* upper bound on the amount of social influence a single individual may have?

To what extent can the maximum social influence of a single individual be minimized by solely modifying the communications network and/or manually defining the social hierarchy?
Acknowledgements

William Regli
Rachel Greenstadt
Sven Koenig
Pragnesh Jay Modi
Robert Laß
Joseph Kopena
Chris Cannon
Peter Bogunovich
Walter Mankowski
Colleen Kavanaugh
Kamelia Aryafar
Tom Plick
Ali Shokoufandeh
Jeremy Johnson
Joseph Macker
Nadya Belov
Moshe Kam
Duc Nguyễn
Marcus McCurdy
Isaac Simmons
Gaurav Naik
Dmitriy Bespalov

Advisor
Committee Member
Mentor
Reviewed a Draft of the Dissertation
Lent a Brain
Making Lab Time Awesome
Helping with the Annoying Minutia
Thank you for your time and attention.

Questions?
The Central Limit Theorem

- For large $m$ and $\ell$ the distribution of the sum of order statistics will fall under a normal distribution.

- Asymptotically: $P \left[ \sum_{i=1}^{m} X_{(n,i)} \leq x \right] \xrightarrow{d} \mathcal{N} \left( m \mathbb{E}[X], \frac{\text{Var}(X)}{m} \right)$,

  $P \left[ \sum_{i=n-\ell+1}^{n} X_{(n,i)} \leq x \right] \xrightarrow{d} \mathcal{N} \left( \ell \mathbb{E}[X], \frac{\text{Var}(X)}{\ell} \right)$.

- For continuous uniform distributions the quotient of the expected values of the nominator and denominator in $Z$ should give a relatively unbiased estimation of the true expected value of $Z$, which itself should have very low variance.

- We have shown that the same is true for a number of other distributions, and by the central limit theorem this holds in general.

- Therefore,

  $\mathbb{E}[Z] = \frac{\ell \mathbb{E}[X]}{m \mathbb{E}[X]} = \frac{\ell}{m}$.
Experimental Setup
(Art Gallery)

- $n = \#\text{ Agents}$
- A series of $n$-gons randomly generated by connecting $n$ uniformly distributed vertices in the unit square of the Cartesian plane according to the “Two Peasants” method.
- 32 random polygons created for each value of $n$.
- Agent instantiated at each vertex of each randomly generated polygon and the algorithm run.
- The optimal dominating set was also calculated.
Randomly place a set of points on the plane.
Draw a line between the two points with extremal $x$ values...
“Two Peasants” Point Set Polygonization

...dividing the plane into half-spaces.
“Two Peasants” Point Set Polygonization

In each half-space, the vertices are connected to each other in order of increasing $x$ value.
“Two Peasants” Point Set Polygonization

Resulting in a random, simple polygon.
Theorem

Selecting nodes in parallel will not induce a cycle in the constrained forest.

Proof sketch.

1. Any cycle implies that at least two edges have the same potential.
2. If two edges have the same potential, ties are broken based upon the ordering of vertices.
3. Thus, if there is a cycle, either two vertices have the same unique ID or the graph is not simple \(\implies\).
Theorem

The vector $\mathbf{y}$ is dual-feasible and has the property

$$\sum_{e \in H_T} w(e) \leq \sum_{e \in H_T} \sum_{S : e \in \delta(S)} y_s.$$ 

Proof sketch.

1. Feasibility follows from the path-cost update (based on the potential function).
2. The fact that all open fringe vertices are updated with the path-cost ensures the property.
Theorem

If there exists an $\tilde{\omega}$ such that all edge weights are in the range $[\tilde{\omega}, \frac{3}{2}\tilde{\omega}]$ then the cost of the final solution is bounded above by $(2 - \frac{2}{|V|})Z^*_IP$.

Proof sketch.

1. The average degree of a vertex in a forest of at most $n$ vertices is at most $2 - \frac{2}{n}$.
2. Since the degree of each vertex in the forest is bounded, we can bound the total amount of edge weight that is added each round.
3. The proof then follows by induction.
Theorem

The solution will be 2-optimal if $\forall e = \langle u, v \rangle \in \tilde{E}$:

1. $e$ was not added mutually and at most one additional edge incident to $u$ becomes tight in a later round; or
2. $e$ was added mutually and at most three additional edges incident to $u$ become tight in a later round.

Proof sketch.

This is a direct consequence of maintaining the invariant

$$\sum_{e \in H_\tau} w(e) \leq 2 \sum_{S \subset V} y_S.$$
Theorem

If all component unions are mutual, $\alpha \geq \frac{n}{2}$, and all agents can communicate with each other in $O(1)$ rounds, then the algorithm will quiesce in $O(\log n)$ rounds.

Proof sketch.

Let $A_f(t) = A(t)$ be an upper bound on the number of active components at the beginning of round $t$. Let $L_f(t) = L(t)$ be an upper bound on the total number of components at the beginning of round $t$.

$$A(t) = \max \left( \frac{A(t-1)}{2}, \min \left( A(t-1), L(t-1) - A(t-1) \right) \right)$$

$$L(t) = L(t-1) - \frac{A(t-1)}{2}.$$ 

The $A(t)$ recurrence will converge exponentially.
**Theorem**

The algorithm will run in worst case linear communication rounds with respect to the number of vertices.

**Proof sketch.**

The final forest has at most $n - 1$ edges, and at least one vertex is added during each round.
Experimental Setup

- Generate ~100k random graphs of varying edge density and size.
  - Density \( \in \{0.1, 0.2, \ldots, 1.0\} \).
  - Number of vertices \( \in \{1, 2, \ldots, 100\} \).
  - \( \min\{2^n, 100\} \) random graphs for each combination.

- For each graph, generate a hierarchy using both DFS and Mobed.

- Randomly add a new vertex to each graph (maintain edge density) and generate a new hierarchy using both DFS and Mobed.

- Record runtime of this step.
- Record the *edit distance* between the hierarchies.
Runtime for an Agent Addition:

Theory is confirmed: hierarchy modifications run in worst-case linear time. Works best on sparse graphs.

[Graph showing runtime vs. number of agents for various densities (0.1 to 1.0), with linear time growth.]
Comparison of Edit Distance:

Mobed has up to a 300% better edit distance than DFS for sparse graphs. For dense graphs, Mobed is never more than 2 edits worse.
Comparison of Edit Distance
for Barabási-Albert small world graphs

-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Edit Distance
Small World Graph Density

Return to the Mobed Summary

E. A. Sultanik (Drexel University)
Dynamic Agent Organizations
2010-09-08
A18 / A23
Example: Encoding Steiner Forest

Whether or not an edge $e$ will be in the forest: $x_e \in \{0, 1\}$.

$$\min \sum_{e \in E} w(e)x_e$$

s.t. $$\sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset$$

$$x_e \geq 0, \quad \forall e \in E,$$

$$\max \sum_{S \subset V} f(S)y_S$$

s.t. $$\sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E$$

$$y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.$$

M. Aggarwal and N. Garg
A Scaling Technique for Better Network Design.
Example: Encoding Steiner Forest

The weight of edge $e$: $w(e)$.

$$\min \sum_{e \in E} w(e) x_e$$

s.t. $$\sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset$$

$$x_e \geq 0, \quad \forall e \in E,$$

$$\max \sum_{S \subset V} f(S) y_S$$

s.t. $$\sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E$$

$$y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.$$
Example: Encoding Steiner Forest

\[ f(S) = 1 \text{ iff } \emptyset \neq S \cap \{1, 2, 4\} \neq \{1, 2, 4\} \]

\[
\begin{align*}
\text{min } & \sum_{e \in E} w(e)x_e \\
\text{s.t. } & \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \\
& x_e \geq 0, \quad \forall e \in E,
\end{align*}
\]

\[
\begin{align*}
\text{max } & \sum_{S \subset V} f(S)y_S \\
\text{s.t. } & \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \\
& y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.
\end{align*}
\]

M. Aggarwal and N. Garg
A Scaling Technique for Better Network Design.
Example: Encoding Steiner Forest

Each variable in the primal becomes a constraint in the dual

\[
\begin{align*}
\text{min} \quad & \sum_{e \in E} w(e)x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \\
& x_e \geq 0, \quad \forall e \in E, \\
\end{align*}
\]

\[
\begin{align*}
\text{max} \quad & \sum_{S \subset V} f(S)y_S \\
\text{s.t.} \quad & \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \\
& y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.
\end{align*}
\]

M. Aggarwal and N. Garg
A Scaling Technique for Better Network Design.
Example: Encoding Steiner Forest

\[
\begin{align*}
\min \quad & \sum_{e \in E} w(e) x_e \\
\text{s.t.} \quad & \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \\
\quad & x_e \geq 0,
\end{align*}
\]

\[
\begin{align*}
\max \quad & \sum_{S \subset V} f(S) y_S \\
\text{s.t.} \quad & \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \\
\quad & y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.
\end{align*}
\]

Each constraint in the primal becomes a variable in the dual.

M. Aggarwal and N. Garg
A Scaling Technique for Better Network Design.
Example: Encoding Steiner Forest

This is a mechanical process!

\[ \begin{align*}
\min & \quad \sum_{e \in E} w(e)x_e \\
\text{s.t.} & \quad \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \\
& \quad x_e \geq 0, \quad \forall e \in E,
\end{align*} \]

\[ \begin{align*}
\max & \quad \sum_{S \subset V} f(S)y_S \\
\text{s.t.} & \quad \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \\
& \quad y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset.
\end{align*} \]
Example: Encoding Steiner Forest

\[ \min \sum_{e \in E} w(e) x_e \]
\[ \text{s.t. } \sum_{e \in \delta(S)} x_e \geq f(S), \quad \forall S \subset V : S \neq \emptyset \]
\[ x_e \geq 0, \quad \forall e \in E, \]

\[ \max \sum_{S \subset V} f(S) y_S \]
\[ \text{s.t. } \sum_{S : e \in \delta(S)} y_S \leq w_e, \quad \forall e \in E \]
\[ y_S \geq 0, \quad \forall S \subset V : S \neq \emptyset. \]
Proper Functions

\[
\begin{align*}
\min \sum_{e \in E} w(e)x_e & \quad \text{max} \sum_{S \subseteq V} f(S)y_S \\
\text{s.t.} \sum_{e \in \delta(S)} x_e & \geq f(S), \quad \forall S \subseteq V : S \neq \emptyset & \text{s.t.} \sum_{S : e \in \delta(S)} y_S & \leq w_e, \quad \forall e \in E \\
x_e & \geq 0, \quad \forall e \in E, & y_S & \geq 0, \quad \forall S \subseteq V : S \neq \emptyset.
\end{align*}
\]

A function on the powerset of a set of vertices, \( f : 2^V \rightarrow \{0, 1\} \), is said to be proper if the following are true:

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Null</strong></td>
<td>( f(\emptyset) = 0 )</td>
</tr>
<tr>
<td><strong>Symmetry</strong></td>
<td>( \forall S \subseteq V : f(S) = f(V - S) )</td>
</tr>
<tr>
<td><strong>Disjointness</strong></td>
<td>( \forall A, B \subseteq V : (A \cap B = \emptyset) \Rightarrow f(A \cup B) \leq \max{f(A), f(B)} ).</td>
</tr>
</tbody>
</table>
If \( f \) is proper then the \textbf{sequential} algorithm will.

- ...run in polynomial time; and
- ...produce a solution that is \( 2\)-\textit{OPT} (\( i.e. \), the cost will be no more than two times the cost of the optimal solution).

“\textit{Constrained Forest Problems}”

- Many constrained forest problems are \textbf{NP-HARD}.
- Various extensions (\( e.g. \), supermodular, well spaced, &c.).