Intelligence Surveillance and Reconnaissance Asset
Assignment for Optimal Mission Effectiveness

Capt Ryan Kappedal, M.S.
Maj August Roesener, Ph.D.
Maj Shane Hall, Ph.D.

Sponsor
USSTRATCOM
**Title:** Intelligence Surveillance and Reconnaissance Asset Assignment for Optimal Mission Effectiveness

**Performing Organization:** Air Force Institute of Technology

**Abstract:**
**Problem Statement**

- USSTRATCOM has requested assistance assigning sensors to multi-stage missions
- Each mission has several stages, and some sensors may be shared between missions
- Each sensor has a distinct probability of success at a unique mission’s stage
  - These probabilities are not always known until just prior to tasking
Problem Example

Set of sensors

Set of missions
This research seeks to

- Define $\mathbf{x} \in \mathbb{B}^n$, $\mathbf{x} \in \Omega$, and $F(\mathbf{x})$
- Find techniques to Maximize $F(\mathbf{x})$
Reliability Theory

- Basic bridge structure network
  - Assuming independent failures of elements, calculations are simple

\[ R = [1 - (1 - P(E_{11}))(1 - P(E_{21})))][P(E_{12})] \]
• General bridge network
  – System is called a series-parallel redundant system

\[
R = \prod_{j=1}^{m} \left( 1 - \prod_{i=1}^{n} (1 - P(E_{ij})) \right)
\]
Simple Resource Allocation

\[
\max_{j=1}^{n} \sum_{j=1}^{n} f_j(x_j),
\]

subject to

\[
\sum_{j=1}^{n} x_j \leq b
\]

\[x \in \mathbb{Z}^n\]

- \(f_j(x_j)\) is concave, increasing, and non-linear
- Problem is NP-Complete
Weighted Sum Scalarization

\[ \max_{x \in \Omega} \sum_{k=1}^{p} \lambda_k f_k(x), \]
\[ x \in \mathbb{B}^n \]
\[ f(x) \in \mathbb{R} \]

• Converts a multi-objective function into a single objective function
• Answers are guaranteed to be efficient
Methodology

• Formulation of Maximum Utility Sensor Assignment Problem (MUSAP) as an IP
• Solution techniques
  – Explicit enumeration
  – Heuristics (Simulated Annealing)
Model Formulation

\[ x_{ijk} \equiv \begin{cases} 
1 & \text{if sensor } i \text{ is assigned to mission } j, \text{ at stage } k \\
0 & \text{otherwise.} 
\end{cases} \]

- Decision Variables
  - Current formulation assumes any sensor can do any mission at any stage
  - Each assignment has a probability of success, \( \rho_{ijk} \)
Building the Objective Function

- General bridge structure network for a single mission
  - Assumes independence of stages and sensors
Building the Objective Function

Probability of success at an individual stage

\[ 1 - \prod_{i=1}^{\alpha} (1 - \rho_{ijk} x_{ijk}) \]

Probability of success at an individual mission

\[ \prod_{k=1}^{\chi} \left( 1 - \prod_{i=1}^{\alpha} (1 - \rho_{ijk} x_{ijk}) \right) \]

Mission probabilities aggregated into a weighted-sum scalarized objective function

\[ \sum_{j=1}^{\beta} w_j \left( \prod_{k=1}^{\chi} \left[ 1 - \prod_{i=1}^{\alpha} (1 - \rho_{ijk} x_{ijk}) \right] \right) \]
Constraining the Space

\[
\sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} x_{ijk} \leq \alpha \quad \forall \ k = 1, 2, \ldots, \chi
\]

Do not assign more sensors than are available at a given stage

\[
\sum_{j=1}^{\beta} x_{ijk} \leq 1 \quad \forall \ i = 1, 2, \ldots, \alpha, \ k = 1, 2, \ldots, \chi
\]

Do not assign a sensor to more than one mission at a given stage
Complete Formulation of MUSAP

\[
\max \sum_{j=1}^{\beta} w_j \left( \prod_{k=1}^{\chi} \left[ 1 - \prod_{i=1}^{\alpha} (1 - \rho_{ijk} x_{ijk}) \right] \right)
\]

subject to:

\[
\sum_{i=1}^{\alpha} \sum_{j=1}^{\beta} x_{ijk} \leq \alpha \quad \forall \quad k = 1, 2, \ldots, \chi
\]

\[
\sum_{j=1}^{\beta} x_{ijk} \leq 1 \quad \forall \quad i = 1, 2, \ldots, \alpha, \quad k = 1, 2, \ldots, \chi
\]

\[
x_{ijk} \in \{0, 1\} \quad \forall \quad i = 1, 2, \ldots, \alpha, \quad j = 1, 2, \ldots, \beta, \quad k = 1, 2, \ldots, \chi
\]
## Intractability of Test Problems

<table>
<thead>
<tr>
<th>sensors</th>
<th>missions</th>
<th>stages</th>
<th>number of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>4</td>
<td>$1.05 \times 10^6$</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>4</td>
<td>$2.82 \times 10^8$</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>4</td>
<td>$1.15 \times 10^{18}$</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>4</td>
<td>$8.37 \times 10^{22}$</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>4</td>
<td>$2.37 \times 10^{32}$</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>4</td>
<td>$4.64 \times 10^{39}$</td>
</tr>
<tr>
<td>40</td>
<td>16</td>
<td>4</td>
<td>$1.46 \times 10^{48}$</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>4</td>
<td>$7.70 \times 10^{57}$</td>
</tr>
</tbody>
</table>
Heuristic Techniques

• Construction Heuristics
  – Greedy Individual and Greedy Marginal

• Local Search
  – Simulated Annealing (with several parameter settings)

• Combining Greedy Construction algorithms with Local search is a method called GRASP (Feo and Resende 1989)
Neighborhoods

\[ f(x) \]

\[ x \]

\[ n(1) \]

\[ n(2) \]
## Experimental Settings

### Experimental Factors

<table>
<thead>
<tr>
<th>neighborhood</th>
<th>$\gamma$</th>
<th>starting point</th>
<th>diversify after</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-swap</td>
<td>0 (Simple Ascent)</td>
<td>marginal</td>
<td>50%</td>
</tr>
<tr>
<td>2-swap</td>
<td>0.9 (Simulated Annealing)</td>
<td>individual</td>
<td>75%</td>
</tr>
<tr>
<td>3-swap</td>
<td>0.8 (Simulated Annealing)</td>
<td>-</td>
<td>never</td>
</tr>
<tr>
<td>-</td>
<td>0.7 (Simulated Annealing)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- 72 total algorithms
  
  $$(3 \times 4 \times 2 \times 3 = 72)$$
Full Enumeration

- Provides a baseline from comparison of heuristics techniques
- Generally takes a large number of function evaluations
  - Every point must be examined
- Not practical for implementation
Simulated Annealing Implementation

<table>
<thead>
<tr>
<th>dist</th>
<th>50%</th>
<th>75%</th>
<th>never</th>
<th>1-swp</th>
<th>2-swp</th>
<th>3-swp</th>
<th>0.9</th>
<th>0.8</th>
<th>0.7</th>
<th>0</th>
<th>Mrg</th>
<th>Ind</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>94.9</td>
<td>94.8</td>
<td>94.9</td>
<td>91.2%</td>
<td>95.6%</td>
<td>97.9%</td>
<td>99.0%</td>
<td>96.6%</td>
<td>95.0%</td>
<td>88.9%</td>
<td>95.0%</td>
<td>94.8%</td>
</tr>
<tr>
<td>5%</td>
<td>86.6</td>
<td>86.9</td>
<td>87.8</td>
<td>82.9%</td>
<td>89.1%</td>
<td>89.4%</td>
<td>88.4%</td>
<td>88.2%</td>
<td>87.2%</td>
<td>84.7%</td>
<td>86.9%</td>
<td>87.4%</td>
</tr>
<tr>
<td>1%</td>
<td>12.9</td>
<td>14.2</td>
<td>17.4</td>
<td>16.1%</td>
<td>17.8%</td>
<td>10.6%</td>
<td>9.8%</td>
<td>14.0%</td>
<td>15.3%</td>
<td>20.1%</td>
<td>13.5%</td>
<td>16.1%</td>
</tr>
</tbody>
</table>

Table B.3  Aggregate results for problem size $10 \times 4 \times 4$

<table>
<thead>
<tr>
<th>dist</th>
<th>50%</th>
<th>75%</th>
<th>never</th>
<th>1-swp</th>
<th>2-swp</th>
<th>3-swp</th>
<th>0.9</th>
<th>0.8</th>
<th>0.7</th>
<th>0</th>
<th>Mrg</th>
<th>Ind</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>7.1</td>
<td>7.3</td>
<td>7.4</td>
<td>3.8%</td>
<td>8.2%</td>
<td>9.7%</td>
<td>9.6%</td>
<td>5.0%</td>
<td>3.4%</td>
<td>10.9%</td>
<td>6.7%</td>
<td>7.7%</td>
</tr>
<tr>
<td>5%</td>
<td>0.4</td>
<td>0.4</td>
<td>0.5</td>
<td>0.1%</td>
<td>0.5%</td>
<td>0.6%</td>
<td>0.5%</td>
<td>0.2%</td>
<td>0.2%</td>
<td>0.8%</td>
<td>0.3%</td>
<td>0.5%</td>
</tr>
<tr>
<td>1%</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table B.5  Aggregate results for problem size $10 \times 7 \times 4$

Sensor sparse networks have much worse results
Initial Results

- 10 x 4 x 4 network does well
- 10 x 7 x 4 network does poorly
  - Even with 7x the iterations
- Re-evaluating the SA algorithm
  - If the last sensor is removed from any individual mission’s stage, the mission fails
  - A failed mission is very difficult to recover to a successful stage
  - Many local optima are created
  - Move evident in “sparse” vs. “saturated” networks
- Modification prevents “auto-fails” the every mission and stage has at least one sensor
  - This eliminates the “greedy individual constructions”
  - Only 36 algorithms to consider
## Modified Algorithm (Saturated Network)

### Aggregate Results

<table>
<thead>
<tr>
<th>dist</th>
<th>Diversification Rule</th>
<th>Neighborhood</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50%</td>
<td>75%</td>
<td>never</td>
</tr>
<tr>
<td>10%</td>
<td>99.6%</td>
<td>99.7%</td>
<td>99.7%</td>
</tr>
<tr>
<td>5%</td>
<td>91.1%</td>
<td>91.6%</td>
<td>92.2%</td>
</tr>
<tr>
<td>1%</td>
<td>12.2%</td>
<td>14.2%</td>
<td>17.1%</td>
</tr>
</tbody>
</table>

Table B.7  Aggregate results for problem size 10\( \times \)4\( \times \)4: Algorithm Modification

### Aggregate Results

<table>
<thead>
<tr>
<th>dist</th>
<th>Diversification Rule</th>
<th>Neighborhood</th>
<th>( \gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50%</td>
<td>75%</td>
<td>never</td>
</tr>
<tr>
<td>10%</td>
<td>60.5%</td>
<td>60.2%</td>
<td>61.3%</td>
</tr>
<tr>
<td>5%</td>
<td>21.5%</td>
<td>21.7%</td>
<td>24.4%</td>
</tr>
<tr>
<td>1%</td>
<td>0.6%</td>
<td>0.8%</td>
<td>1.2%</td>
</tr>
</tbody>
</table>

Table B.9  Aggregate results for problem size 10\( \times \)7\( \times \)4: Algorithm Modification
Simulated Annealing Convergence

10x4x4 Network

10x7x4 Network
Overall Heuristics Observations

• Algorithm Type and Cooling Schedule
  – SA with a cooling schedule of 0.9 produces the highest quality results only in the sparse networks
  – In saturated networks, the Simple Ascent algorithm is more effective at returning high quality solutions.
  – May be because sensor saturated networks have far less local optimal than the sensor sparse networks
Overall Heuristics Observations

• Neighborhood Functions
  – 1-swap and 2-swap neighborhoods proved to be the best neighborhoods by consistently performing well in the test problems
  – Larger neighborhoods performed very poorly, especially in the larger problem sizes
  – In the larger neighborhoods, there are many different points to search with higher n-swap neighborhoods such that the probability of finding improving moves is smaller
  – In this case, the simpler methods are better.
• Diversification Rules
  – “Never diversify” is the best rule
  – Implication for MUSAP is that starting off with a strategy and utilizing it throughout, the algorithm performs much better than attempting to switch midstream.
  – Does not rule out the possibility of using a different type of diversification strategy that doesn't use the "switch after certain percentage of iterations" rule.
Conclusions

• This research has formulated the USSTRATCOM's sensor assignment problem as a type of resource allocation problem.
  – Shown the utility of simulated annealing and simple ascent (iterative improvement) to solve that formulation.
  – As the problems became more complex, simulated annealing with geometric cooling schedules emerged as the most effective algorithm
Recommended Future Research

- Inserting Dependant Probabilities
  - Another effort has created this function
- Simulated Annealing Parameter Specification
  - 10x outer loop / inner loop ratio could be changed
- Varying Weights
  - Mission Drop thresholds
- Other Heuristics Techniques
  - GA has been successful in the single variable case