Task Assignment in a Human-Autonomous Image Labeling System

Addison Bohannon
Applied Math, Statistics, & Scientific Computation

Advisors:

Vernon Lawhern
Army Research Laboratory

Brian Sadler
Army Research Laboratory

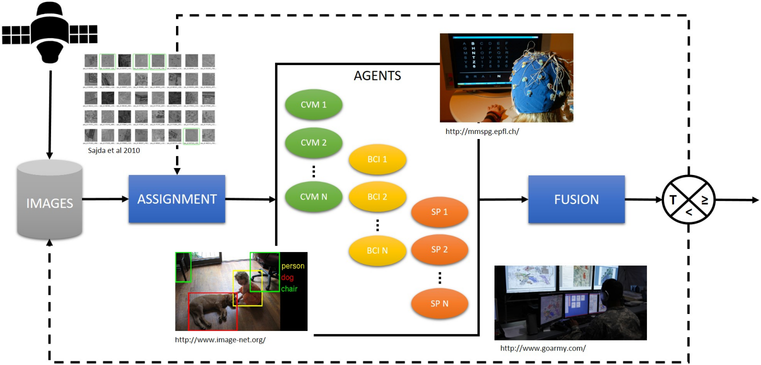
December 9, 2015

Outline

1. Problem
2. Approach
    - Branch and Bound
    - Bounding Function
3. Results
    - Accuracy
    - Time Complexity
    - Computational Efficiency
4. Updates
5. Supplement

Human-Autonomous Image Labeling System

OSD Autonomy Research Pilot Initiative, Army Research Laboratory



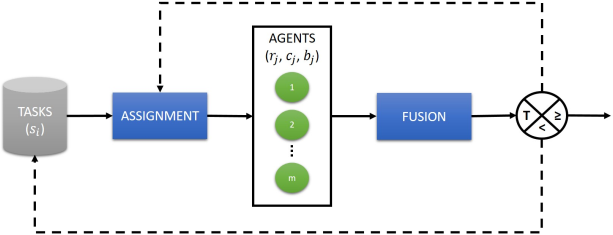
Sajda et al 2010

http://mmspg.epfl.ch/

http://www.image-net.org/

http://www.goarmy.com/

Iterative Task Assignment System



Problem Statement

1. Assignment Problem – How to optimally assign homogeneous binary classification tasks amongst diverse agents?
2. Joint Classification Problem – How to dynamically combine multiple labels from noisy agents without supervised knowledge?

Generalized Assignment Problem (GAP)

Morales and Romeijn [2004]; Kundakcioglu and Alizamir [2008]

$$Z = \max_{\mathbf{x}} \sum_{i \in I} \sum_{j \in J} v_{ji} x_{ji} \qquad (1)$$

1. $\sum_{i \in I} c_{ji} x_{ji} \leq b_j, \; j \in J$
2. $\sum_{j \in J} x_{ji} = 1, \; i \in I$
3. $x_{ji} \in \{0, 1\}$
4. $c_{ji}, b_{ji} \in \mathbb{Z}_+$
5. $v_{ji} = g(r_j, s_j) \geq 0$

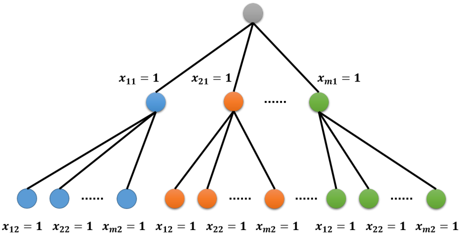
- n – number of tasks
- m – number of agents
- x_{ji} – assignment of task i to agent j
- v_{ji} – assignment value of task i to agent j
- c_{ji} – assignment cost of task i to agent j
- b_j – budget for agent j
- r_j – reliability of agent j
- s_i – classification confidence of task i

Branch and Bound Algorithm

Fisher [2004]; Morales and Romeijn [2004]; Kundakcioglu and Alizamir [2008]

Pseudo-code: Branch and Bound

```
Data: v, c, b
Result: $x, Z$
$Z = Z_0, \ queue = p_0$;
while $queue \neq \emptyset$ do
    1. Select $p \in queue$
    2. Branch on $p$
    3. for $j = 1, \dots, m$ do
        Bound $p_j$
    end
    4. if $Z_j > Z$ then
        if $x_j$ is feasible then
            $x = x_j, \ Z = Z_j$
        else
            add $p_j$ to $queue$
        end
    end
end
```



Lagrangian Relaxation

Fisher [2004]; Fisher *et al.* [1986]; Boyd and Vandenberghe [2004]

We relax the semi-assignment constraint, [2], in (1):

$$L^a(\lambda) = \max_{\mathbf{x}} \left(\sum_{i \in I} \sum_{j \in J} v_{ji} x_{ji} + \sum_{i \in I} \lambda_i \left(1 - \sum_{j \in J} x_{ji} \right) \right) \tag{2}$$

1. $\sum_{i \in I} c_{ji} x_{ji} \leq b_j, \; j \in J$
2. $x_{ji} \in \{0, 1\}$
3. $c_{ji}, b_{ji} \in \mathbb{Z}_+$

which yields m distinct 0-1 knapsack problems for fixed λ:

$$L_j^a(\lambda) = \max_{\mathbf{x}} \left(\sum_{i \in I} (v_{ji} - \lambda_i) \, x_{ji} \right), \; j \in J, \; s.t. \; [1, 2, 3], \tag{3}$$

and the dual problem provides a bounding function:

$$Z_{Da} = \min_{\lambda} L^a(\lambda) = \min_{\lambda} \left(\sum_{j \in J} L_j^a(\lambda) + \sum_{i \in I} \lambda_i \right) \geq Z. \tag{4}$$

Sub-gradient Method

Fisher [2004]; Boyd and Vandenberghe [2004]

Although Z_{Da} is not everywhere differentiable, a sub-gradient descent method can be implemented. A subgradient of a function, f at t_0 is a vector, ν, such that

$$f(t) \leq f(t_0) + \nu(t - t_0), \ \forall \ t. \tag{5}$$

$\mathbf{g}^k = (1 - \sum_j x_{j1}, \ldots, 1 - \sum_j x_{jn})$ is a sub-gradient of

$$L^a(\lambda^k) = \max_{\mathbf{x}} \left(\sum_{i \in I} \sum_{j \in J} v_{ji} x_{ji} + \sum_{i \in I} \lambda_i^k \left(1 - \sum_{j \in J} x_{ji} \right) \right)$$

at λ_k, and we can use the following iterative step for the sub-gradient descent algorithm:

$$\lambda_i^{k+1} = \lambda_i^k - \alpha_k g_i^k. \tag{6}$$

Sub-gradient Method

Fisher [2004]; Boyd and Vandenberghe [2004]

Algorithm: Subgradient Method

```
Data: $\mathbf{v}_j = (v_{j1}, \ldots, v_{jn})^T, \mathbf{c}_j = (c_{j1}, \ldots, c_{jn})^T, b_j, \lambda^0$
Result: $\mathbf{x}, Z$
$k = 0$;
while convergence condition is not met do
    for $j = 1, \ldots, m$ do
        $[\mathbf{x}_j, Z_j] = knapsack(\mathbf{v}_j - \lambda^k, \mathbf{c}_j, b_j)$;
    end
    for $i = 1, \ldots, n$ do
        $\lambda_i^{k+1} = \lambda_i^k - \alpha_k(1 - \sum_{j \in J} \mathbf{x}_{ji})$;
    end
    $k = k + 1,\ Z = \sum_j Z_j + \sum_i \lambda_i^k$;
end
```

0-1 Knapsack Problem

Otten

The 0-1 knapsack problem,

$$L_j^a(\lambda) = \max_{\mathbf{x}} \left(\sum_{i \in I} \left(v_{ji} - \lambda_i \right) x_{ji} \right), \; j \in J$$
$$= \max_{\mathbf{x}} \left(\sum_{i \in I} v_{ji}^* x_{ji} \right) \qquad (7)$$

1. $\sum_{i \in I} c_{ji} x_{ji} \leq b_j,$
2. $x_{ji} \in \{0, 1\}$
3. $c_{ji}, b_{ji} \in \mathbb{Z}_+$

has a pseudo-polynomial time dynamic programming algorithm.

0-1 Knapsack Problem

Otten

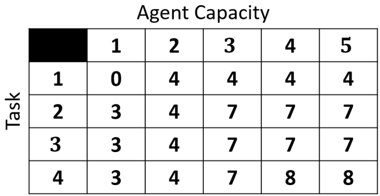
Algorithm: 0-1 Knapsack Problem

```
Data: $\mathbf{v}_j^* = (v_{j1}^*, \ldots, v_{jn}^*)^T, \mathbf{c}_j = (c_{j1}, \ldots, c_{jn})^T, b_j$
Result: $\mathbf{x}_j = (x_{j1}, \ldots, x_{jn})^T, Z_j$
$M = \{0\}^{n \times b_j},\ S = \{0\}^{n \times b_j}, \mathbf{x}_j = \{0\}^n$;
for $i = 1, \ldots, n$ do
    for $l = 1, \ldots, b_j$ do
        $M(i, l) = \max(M(i-1, j), M(i-1, j - c_j(i)) + v_j^*(i))$;
        if $M(i-1, j - c_j(i)) + v_j^*(i)) > M(i-1, j)$ then
            $S(i, l) = 1$;
        end
    end
end
for $i = n, \ldots, 1$ do
    if $S(i, K)$ then
        $x_j(i) = 1, K = K - c_j(i)$;
    end
end
$Z_j = M(n, b_j)$ ;
```

0-1 Knapsack Problem

Let $\mathbf{v}_j^* = (4, 3, 2, 1)$, $\mathbf{c}_j = (2, 1, 3, 1)$, and $b_j = 5$.

$M =$



Agent Capacity

Task

	1	2	3	4	5
1	0	4	4	4	4
2	3	4	7	7	7
3	3	4	7	7	7
4	3	4	7	8	8

$S =$

Agent Capacity

Task

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	1	1	1
3	0	0	0	0	0
4	0	0	0	1	1

Validation

Problem

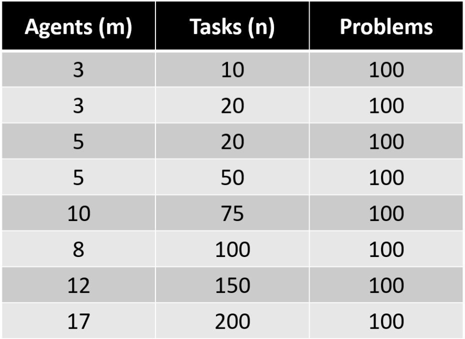
- Randomized problems of sizes from Fisher *et al.* [1986]
- Compare against MATLAB integer programming application
    - NP-hard problem (no “analytical” solution)
    - Compare target function values, Z
    - MATLAB uses Branch and Bound (with plane cutting techniques, integer relaxation)

Implementation

- MATLAB R2015b
- Personal Laptop (8GB, Intel Core i7-4510U, 2.6 GHz, Windows 10-64)

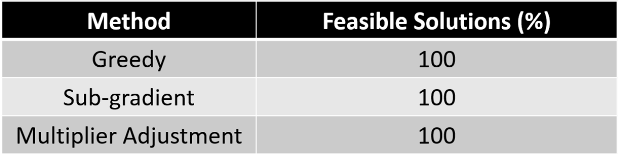
Validation Set-up

- Problem size derived from Fisher *et al.* [1986]
- Randomized **v**, **c**, **b** to facilitate feasible problems



Agents (m)	Tasks (n)	Problems
3	10	100
3	20	100
5	20	100
5	50	100
10	75	100
8	100	100
12	150	100
17	200	100

Accuracy of Methods



Method	Feasible Solutions (%)
Greedy	100
Sub-gradient	100
Multiplier Adjustment	100

Accuracy of Methods

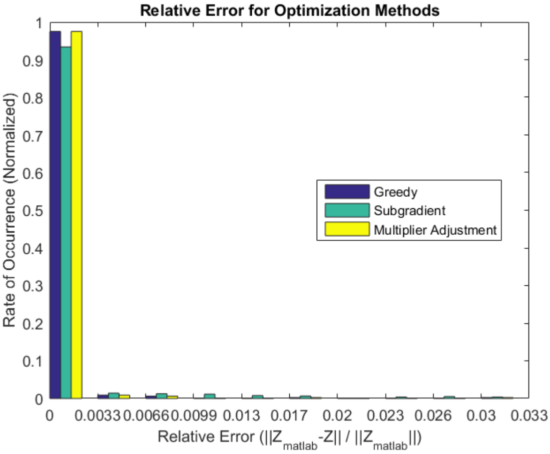
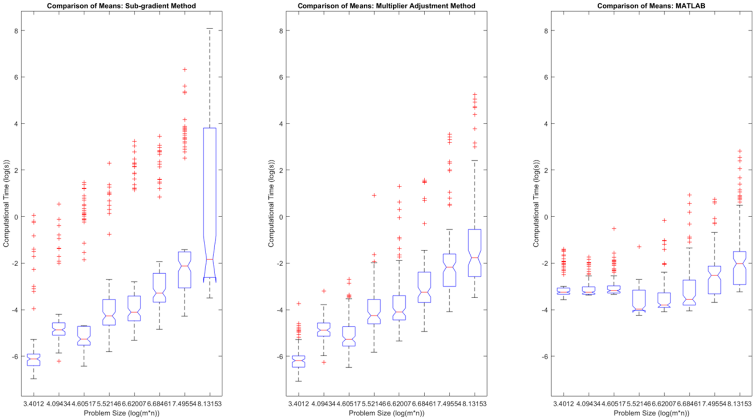


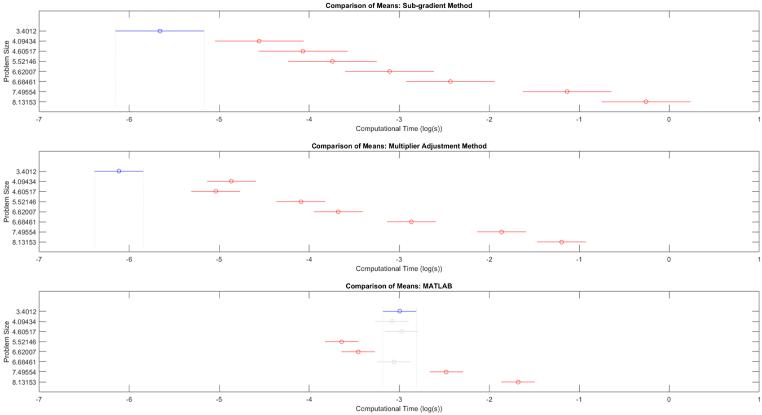
Figure: Target function values, Z, for each method compared against target function value of MATLAB solution. Relative error reported to account for problem size.

Computational Time of Methods



$$F_{subgradient} = 60.29,\ F_{multiplier} = 173.75,\ F_{MATLAB} = 49.03$$

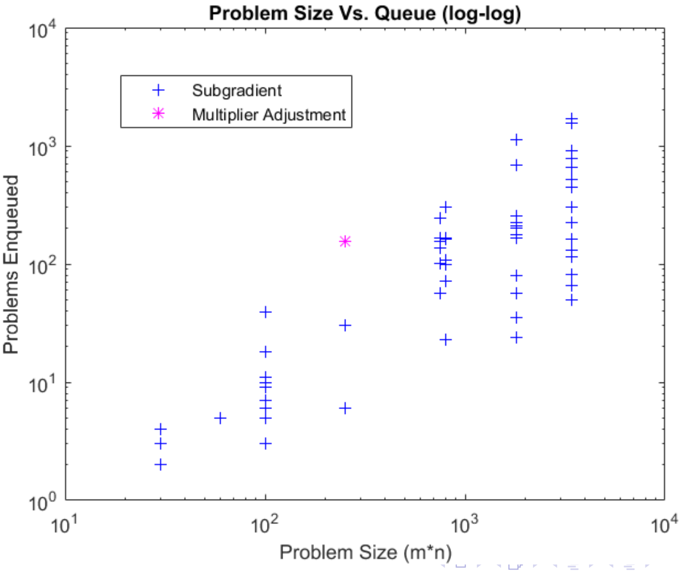
Computational Time of Methods



Computational Time of Methods

Method	Time Complexity $(O((m \times n)^p))$
Greedy	0.75
Sub-gradient	1.0
Multiplier Adjustment	0.96
MATLAB	0.19

Tightness of Bound



Schedule (with Milestones*) - AMSC 663

- Develop Assignment Module (15 OCT - 4 DEC)
  - Implement branch and bound algorithm (6 NOV)*
  - Validate branch and bound algorithm (25 NOV)*
  - Implement greedy search algorithm
  - Mid-year Review (14 DEC)*

Schedule (with Milestones*) - AMSC 664

- Build Image Labeling System (25 JAN - 26 FEB)
  - Build agent classes
  - Develop message-passing framework
  - Integrate all components into a system (26 FEB)*
- Test Image Labeling System (26 FEB - 15 APR)
  - Testing (1 APR)*
  - Performance analysis of test results
- Conclusion (15 APR - 1 MAY)
  - Final Presentation and Results (6 MAY)*

Deliverables

- Software
  - Image Labeling System (fusion module, assignment module, agent classes)
  - Execution script
- Data
  - Office Object Database
  - Office Object RSVP Database
- Analysis
  - Performance analysis of test results
  - Implications for human-autonomous systems

Greedy Method

Efficient implementation of the Branch and Bound algorithm requires a feasible solution to provide a lower bound for solutions:

$$Z_{feasible} \leq Z \leq Z_{Da}. \qquad (8)$$

A tight lower bound requires fewer problems to be enqueued.

Pseudo-code: Greedy Search

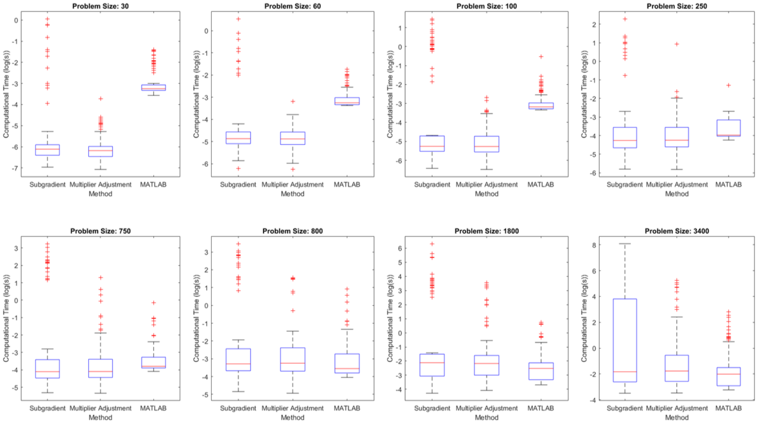
```
Data: $\mathbf{v}, \mathbf{c}, \mathbf{b}$
Result: $\mathbf{x}, Z$
$\mathbf{x} = bound(\mathbf{v}, \mathbf{c}, \mathbf{b})$;
if $\boldsymbol{x}$ is feasible then
    $Z = \mathbf{v}^T\mathbf{x}$, return;
else
    $I_0 = \{i \in I | \sum_{j \in J} x_{ji} = 1\}$;
    for $i \in I_0$ do
        $x_{ji} = 0 \; \forall \; j \in J$;
        1. $sort(v_{ji})$
        2. assign $x_{ji} \; \forall \; j \in J, \; i \in I_0$;
        if $\boldsymbol{x}$ is feasible then
            $Z = \mathbf{v}^T\mathbf{x}$, return;
        end
    end
end
```

Multiplier Adjustment Method

Fisher *et al.* [1986]

```
1. Find $(\mathbf{x}^0, Z^0)$ to (3) s.t. $\sum_j x_{ji} \leq 1$
2. if $\boldsymbol{x}^0$ is feasible then
       return;
   end
3. while $Z^k < Z^{k-1}$ and $\boldsymbol{x}^k$ is not feasible do
       for $i \in \{i \in I | \sum_j x_{ji} = 0\}$ and $j \in J$ do
           Calculate, $\delta_{ji}$, least decrease in $\lambda_i$ for $x_{ji} = 1$
       end
       for $i^* \in \{i \in I | \sum_j x_{ji}^k = 0$ and $\min_2 \delta_{ji} > 0\}$ do
           $\lambda_{i*} = \lambda_{i*} - \min_2 \delta_{ji*}$;
           if possible then
               Find $(\mathbf{x}^k, Z^k)$ to (3) s.t. $\sum_j x_{ji}^k \leq 1$; continue;
           end
       end
   end
```

Computational Time of Methods



Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Marshall L. Fisher, R. Jaikumar, and Luk N. Van Wassenhove. A Multiplier Adjustment Method for the Generalized Assignment Problem. *Management Science*, 32(9):1095–1103, September 1986.

Marshall L. Fisher. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 50(12_supplement):1861–1871, December 2004.

O. Erhun Kundakcioglu and Saed Alizamir. Generalized assignment problem Generalized Assignment Problem. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1153–1162. Springer US, 2008.

Dolores Romero Morales and H. Edwin Romeijn. The Generalized Assignment Problem and Extensions. In Ding-Zhu Du and Panos M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 259–311. Springer US, 2004.

Ralph Otten. Lecture 13: The knapsack problem.