AMSC 664 Final Report: Collaborative Image Triage with Humans and Computer Vision

Addison Bohannon

May 13, 2016

Advisors:

Vernon J. Lawhern
US Army Research Laboratory
Aberdeen Proving Ground, MD 21005 USA
$v$ernon.j.lawhern.civ@mail.mil

Brian M. Sadler
US Army Research Laboratory
Adelphi Laboratory Center, MD 20783 USA
$b$rian.m.sadler6.civ@mail.mil

Abstract

We introduce an image triage system which facilitates the collaboration of human analysts, augmented human analysts, and automated technology to efficiently and accurately classify a two-class database of images. The system iteratively allocates images for binary classification among heterogeneous agents according to the Generalized Assignment Problem (GAP) and combines the classification results using the Spectral Meta-Learner (SML). In simulation, we demonstrate that the proposed system achieves significant speed-up over a naive parallel assignment strategy without sacrificing accuracy.

Contents

1 Introduction

When working with large databases of unlabeled images, there exists a problem of how to efficiently automate the labeling and organization of such data. In the event of sparse training data, the utilization of autonomous algorithms alone leads to over-fitting and poor labeling fidelity. Human analysts may require fewer training images, but the exclusive use of humans may pose a prohibitive time cost. A better system may optimally utilize both the accuracy of human analysts and the speed of intelligent agents for collaborative image triage.

Brain-computer interface (BCI) technologies attempt to improve the performance of a human through augmentation. Rapid Serial Visual Presentation (RSVP), a brain-computer interface paradigm in which brain-signals are recorded from a person while passively viewing images at high rates of speed (2-10 Hz), can be used for high-throughput binary image labeling, but this speed-up comes at the cost of a reduction in labeling accuracy [1].

Sajda, et al. address this problem in [23] through the use a computer vision algorithm to complement the efforts of a human performing image triage via RSVP. The authors explore a serial implementation in which either the computer vision algorithm or RSVP agent first screen the database before the other agent classifies the images. This human-system approach is not unlike the model of multi-class image database labeling presented by Joshi, et al. in [13], where an automated computer vision system prompts a single human to provide binary decisions.

The field of optimal crowd-sourcing addresses the solution of large sets of simple problems such as image triage with human-only ensembles [10–12, 14, 15, 21, 25]. In this paradigm, simple tasks are distributed in parallel to numerous human agents at low-cost (i.e. Amazon Mechanical Turk). Necessary to these approaches are decisions about task allocation and aggregation. In representative allocation approaches, Karger, et al. use a random assignment of tasks with the assumption of agent and task homogeneity [15], and Ho, et al. use the Generalized Assignment Problem (GAP) to determine an optimal allocation of tasks among heterogeneous agents and tasks [10]. As the system can dynamically increase the pool of potential agents, a solution must balance the cost of recruitment with the expected performance of an agent [14]. This prioritizes the ability to infer agent performance without labeled data. Numerous approaches have addressed combining labels of noisy agents and inferring the performance of individual agents from the aggregated responses, of which, the work of Parisi, et al. provides an elegant computational approach to achieve both through the so-called Spectral Meta-Learner (SML) [21].

Here, we present and test a heterogeneous multi-agent image triage system which achieves human-level accuracy while finding a balance in the trade-off between time-cost and accuracy. Our system is designed to utilize three types of independent, heterogeneous agents (computer vision, RSVP, and human), which are assigned images for classification in parallel according to the GAP. Assignment parameters for those agents adapt over multiple iterations of assignment and classification according to the SML. This image triage system (conceptualized in Figure 1) addresses the following:

Accordingly, we want to determine:

1. Image Assignment – How do we optimally assign images in each iteration in order to balance value and time?
2. Joint Classification – How do we infer the label of a task given a set of labels for that task from multiple agents?
3. System Implementation – How to design and develop a flexible system which achieves parallel efficiency?

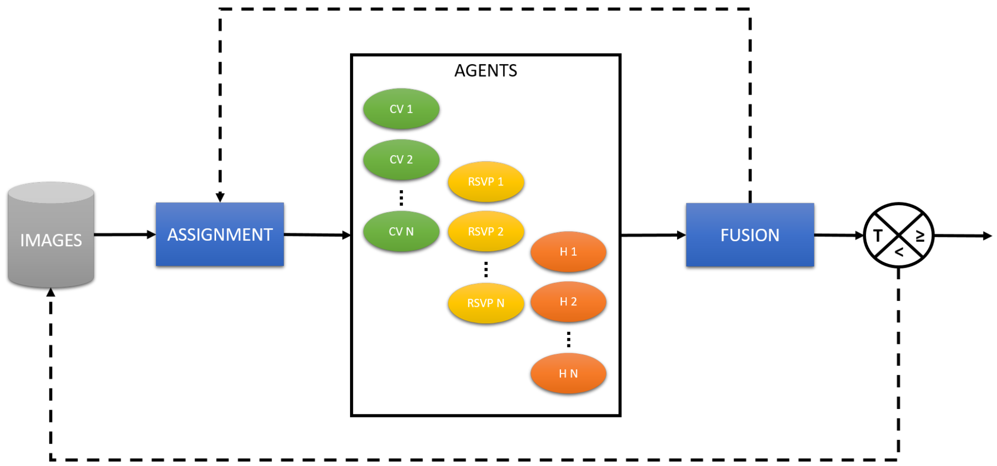


Figure 1: The image triage system. An assignment node distributes images to agents in parallel. The agents perform binary classification, and the results are consolidated at a fusion node. At this point, the confidence in the image classification label is used to threshold images for completion or routing back to the image database for re-assignment. Two forms of feedback occur in the system: the return of images for further assignment and the inference of agent and image properties for further intelligent assignment. Here, agents can be computer vision algorithms, RSVP subjects, or self-paced image analysts.

2 Plan

2.1 Schedule

- Develop assignment module (15 October - 20 December)
  - Implement branch and bound algorithm
  - Validate branch and bound algorithm
  - Implement greedy search algorithm
  - Mid-year presentation
  - Mid-year report
- Revise and update project plan (20 December - 24 January)
- Build image triage System (25 January - 28 February)
  - Build agent classes
  - Develop message-passing framework
  - Integrate all components into a system
- Test image triage system (1 March - 15 April)
  - Testing
  - Performance analysis of test results
- Conclusion (15 April - 13 May)
  - Final presentation
  - Final paper

2.2 Deliverables

- Software

  - Image triage system
  - Execution script
- Data
  - Reduced Caltech101 image database
  - Computer vision models
- Analysis
  - Performance analysis of test results
  - Implications for human-autonomous systems

3 Approach

3.1 Image Assignment

This image assignment problem, where we are looking for the optimal assignment policy over all images and agents, $\{x_{ji}\}_{i\in I,\in J}$, can be mapped onto the Generalized Assignment Problem (GAP) as in [10]. The formulation follows [16]:

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

subject to

1. $\sum_{i\in I} c_{ji}x_{ji} \le b_j,\ j \in J$
2. $\sum_{j\in J} x_{ji} = 1,\ i \in I$
3. $x_{ji} \in \{0,1\}$
4. $c_{ji}, b_j \in \mathbb{Z}_+$
5. $v_{ji} = r_j - s_i + \max_{i^*\in I} s_{i^*}$

Besides having known solutions, GAP captures the important aspects of the assignment problem. Principally, assignment is a decision problem which requires discrete solutions, captured in the 0-1 integer program. Also, the constraints encode the inherent trade-off between assignment value and agent time.

Although GAP is a NP-hard problem, the decision problem is NP-complete [18]. Solving GAP is at least as difficult as any problem in NP and has no known polynomial time solution algorithm, but verifying that a solution is feasible is computable in polynomial time [17]. This ability to verify solutions allows exact solution methods which seek to exhaustively search the solution space for a global maximum.

3.1.1 Branch and Bound Algorithm

Solving GAP with the Branch and Bound algorithm (B&B) is well-established in the literature [8, 16, 18]. B&B is a divide and conquer optimization approach with a bounding function, search strategy, and branching strategy. The algorithm searches branches, or sub-problems of the overall problem, a m-nary tree of height n, and calculates bounds on the sub-problems. Always, a feasible solution is maintained as a lower bound for the optimal solution. This incumbent optimal solution is compared to the upper bound of all sub-problems. For a maximization problem, if the upper bound for a given sub-problem is less than the incumbent optimal solution, then the algorithm can prune these sub-problems and all subsidiary sub-problems. Otherwise, each sub-problem branches into further sub-problems for search. The algorithm iterates until the solution space is exhausted, and a global optimum is found.

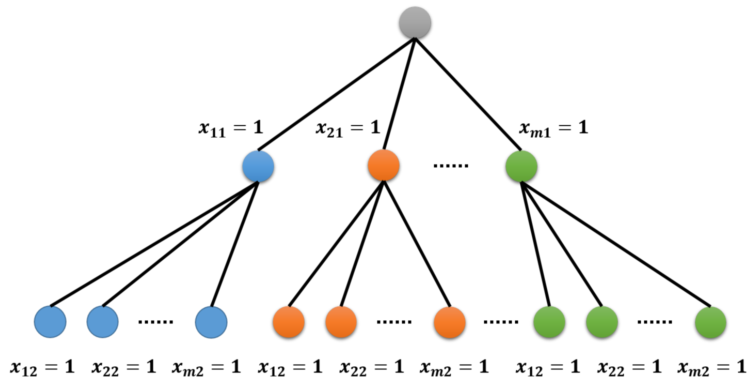


Figure 2: The B&B Algorithm for GAP. The initial candidate problem has no tasks assigned (the grey node). From this problem, the first task can be assigned to each of the m agents, creating m branches. Likewise, for tasks $i = 2, \dots, n$, each task can be assigned to one of the m agents. For each of those sub-problems, an upper bound is calculated and stored if it exceeds the incumbent optimal feasible solution. Here, $x_{ji} = 1$ corresponds to p_j^i in Algorithm 1.

GAP offers an intuitive branching strategy; each image, $i \in I$, can possibly be assigned to any one agent, $j \in J$. This allows a systematic search though each of the n images and branching at each task to create m sub-problems (see Figure 2). For each explored sub-problem, if the upper bound is greater than the incumbent solution, the sub-problem and its upper bound are stored in a running queue. The search strategy consists of selecting the candidate sub-problem from the running queue with the greatest upper bound. For the bounding function, we use Lagrangian relaxation (LR) of constraint [2] of (1) as in [8] and [9]. B&B pseudo-code is shown in Algorithm 1 [4,9].

If all $m \times n$ sub-problems are explored, B&B would be a strictly combinatorial search algorithm. The computational savings arise from sub-problems which are not enqueued. If the upper bound of a sub-problem is less than the incumbent optimal solution, it is not enqueued for later search. For each sub-problem ($x_{ji} = 1,\ i \in I, j \in J$) which is pruned, $\mathcal{O}(m^{(n-1)})$ sub-problems do not require execution of the bounding function. This highlights the importance of a tight bound from the bounding function and a feasible solution to help prune sub-problems throughout B&B.

3.1.2 Lagrangian Relaxation

Considering the set of constraints on GAP, we faced a choice of which constraint subset to relax. Fisher, et al. show decreased computational cost from relaxing constraint [2] over [3] in (1) [9]. Additionally, Morales, et al. prove that the bound from relaxing constraint [2] offers a tighter bound than that of constraint [1] or [3] of (1) [18]. Therefore, we introduced Lagrange multipliers, λ_i, to relax the semi-assignment constraints (constraint [2] in (1)), which results in the following

Algorithm 1: Branch and Bound

```
Data: $Z_0$
Result: $x$
$Z = Z_0,\ queue = p_0$;
while $queue \neq \emptyset$ do
    Select $p^i \in queue$
    for $j \in J$ do
        $Z^i_j = bound(p^i_j)$;
        if $Z^i_j > Z$ then
            if $x_j$ is feasible then
                $x = x^i_j,\ Z = Z^i_j$
            else
                add $p^i_j$ to $queue$
            end
        end
    end
end
```

Lagrangian formulation of (1):

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

The dual formulation,

$$\begin{aligned} d(\boldsymbol{\lambda}) &= \max_{\mathbf{x}} \sum_{j\in J}\left(\sum_{i\in I}(v_{ji} - \lambda_i)\,x_{ji}\right) + \sum_{i\in I}\lambda_i \\ &= \sum_{i\in I}\lambda_i + \sum_{j\in J}\left(\max_{\mathbf{x}}\sum_{i\in I}(v_{ji}-\lambda_i)\,x_{ji}\right), \end{aligned} \tag{3}$$

subject to constraints [1,3-5] of (1), provides a bound on the optimal solution:

$$\min_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}) \geq Z \geq Z_{feasible}, \tag{4}$$

where $Z_{feasible}$ is any feasible solution. Notice that the dual formulation reveals m independent optimization problems for fixed $\boldsymbol{\lambda}$. We use this problem structure to solve the saddle-point problem directly via sub-gradient descent [2]:

$$\begin{cases} \mathbf{x}^{k+1} & = \arg\max_{\mathbf{x}} \sum_{i\in I}\sum_{j\in J}(v_{ji} - \lambda^k_i)x_{ji} \\ & \quad \text{subject to} \quad \sum_{i\in I} c_{ji}x_{ji} \leq b_j \\ \lambda^{k+1}_i & = \lambda^k_i + \alpha_k\left(1 - \sum_{j\in J} x_{ji}\right) \end{cases} \tag{5}$$

We use sub-gradient descent because the dual problem, (3), is not everywhere differentiable. A sub-gradient of a function f at t_0 is a vector, $\nu \in \partial f(t_0)$, such that for t,

$$f(t) \leq f(t_0) + \nu(t - t_0).$$

As with gradient descent, the optimal solution to the dual problem occurs when zero is a sub-gradient of the dual problem [2]. In (5), α_k must satisfy $\lim_{k\to\infty} \alpha_k = 0$ and $\lim_{n\to\infty} \sum_{k=1}^{n} \alpha_k = \infty$. For our implementation, a dynamic step-size was used. The step-size was decreased by a factor of $\gamma < 1$ each time the objective function value remained unchanged between iterations, on the assumption that this indicated an over-shoot of the local minimum. Under these conditions, this iterative procedure guarantees error with convergence on the scale of $\mathcal{O}(\alpha\|g\|^2)$ [19]. The stopping condition was based on the convergence rate for dynamic step sizes. When the bound on the error achieved some relative threshold, the descent algorithm terminated.

The primary advantage of relaxing the semi-assignment constraint is the structure of the dual formulation (3), which yields independent optimization problems for each agent:

$$d_j(\boldsymbol{\lambda}) = \max_{\mathbf{x}} \sum_{i \in I} (v_{ji} - \lambda_i)\, x_{ji}.$$

These problems, known as the 0-1 knapsack problem, can be solved exactly by a pseudo-polynomial time dynamic programming algorithm.

It is known as the knapsack problem because it can be easily visualized as having n items of unique weights and values with a knapsack of fixed weight capacity. The problem is to determine which items if packed will maximize the value in the knapsack. The dynamic programming approach leverages the fact that to determine whether an item is optimally packed for a given capacity, it helps to know the optimal packing list for a problem of a lower capacity. The solution algorithm is shown in Algorithm 2 [20].

Algorithm 2: 0-1 Knapsack

```
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
$Z_j = M(n, b_j)$ ;
```

The complexity of the dynamic programming algorithm is linear in the number of tasks, $\mathcal{O}(nb_j)$, and only grows in complexity with respect to the order of b_j. Since b_j is of sufficiently low order for many problems, the dynamic programming algorithm provides a computationally efficient method for solving the knapsack problem.

3.1.3 Multiplier Adjustment Method

In [9], Fisher, et al. present another approach to solving the dual problem: the multiplier adjustment method. Similar to the steepest descent method, we search the solution space for the single canonical direction in which the smallest decrease in one of the Lagrange multipliers, λ_i, will result in an assignment for a currently unassigned task. Whereas the steps of the sub-gradient descent method often result in large jumps across the solution space–or alternatively, many small steps–these measured descent steps result in a smoother path toward the optimal solution. The multiplier adjustment method remains the standard for comparison of heuristic solution techniques for GAP [16, 18].

We followed the algorithm proposed in [9] for implementation of the multiplier adjustment method. See Algorithm 5 in the appendix for pseudo-code.

3.1.4 Greedy Search

Efficient implementation of B&B requires a feasible solution to provide a lower bound for solutions (see (4)). A tight lower bound requires fewer problems to be enqueued during iterations of the B&B, but it also offers an alternative stand-alone approach to the image assignment problem. In the image labeling system, there will necessarily be a trade-off between speed and optimality of the GAP solution. If a solution method can heuristically provide a good enough solution while saving the cost of an exhaustive B&B search, than it may in fact be the better option. Algorithm 3 shows the greedy search procedure implemented in testing as both a stand-alone method and the initialization of the B&B algorithm. The procedure is based on step (3) in [9].

3.1.5 Validation

Without a known polynomial time algorithm for confirming that an assignment solution is the globally optimal solution, we use the internal MATLAB mixed integer programming function (*intlinprog*()) in the Optimization Toolbox as the known solution for validation results. As the maximum objective value, Z, is not necessarily unique with respect to assignments, $\mathbf{x}$, comparisons are made according to the objective value of the respective solution methods. The methods were implemented in MATLAB R2015b. Validation of the task assignment module ran on a Windows-based laptop computer with an Intel Core i7 2.6 GHz processor and 8GB RAM.

In [9], the results of multiple implementations of B&B were compared for various GAP problems, and these problem sizes are adopted here. A summary of validation problem sizes is shown in Table 1.

We have implemented three distinct methods for solving GAP. The first two are variations of B&B, and thus exact: the sub-gradient descent method ((5)) and the multiplier adjustment method (Algorithm 5); the third, heuristic: the greedy method (Algorithm 3). Although solutions are not verifiable, we can ensure that solutions strictly satisfy the problem constraints of (1). The B&B methods strictly enforce the constraints and guarantee a solution for a feasible problem. This is reflected in experimental results (see Table 2). The greedy method strictly enforces the constraints, but is not guaranteed to return a solution. These cases are reflected in the table as infeasible solutions. Notably, MATLAB failed to find solutions which satisfied the problem constraints on almost all problems. The MATLAB solver uses B&B with the integer constraint relaxed, and only enforces the integer constraint (constraint [3] in (1)) within a tolerance of $\mathcal{O}(10^{-6})$ and the

Algorithm 3: Greedy Search

```
Data: $\mathbf{v}, \mathbf{c}, \mathbf{b}$
Result: $\mathbf{x}, Z$
for $j \in J$ do
    $[\mathbf{x}_j, Z_j] = knapsack(\mathbf{v}_j, \mathbf{c}_j, b_j)$;
end
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
```

Agents (m)	Tasks (n)	Problems
3	10	25
3	20	25
5	20	25
5	520	25
10	75	25
8	100	25
12	150	25
17	200	25

Table 1: Problem sizes of GAP for validation of B&B algorithm implementation. For each problem size, assignment values, $\mathbf{v}$, assignment costs, $\mathbf{c}$, and agent budgets, $\mathbf{b}$ were generated randomly to accommodate feasible problems ($v_{ji} \in \mathbb{R} \sim unif(1,5)$, $c_{ji} \in \mathbb{Z}_+ \sim unif(1,5)$, and $b_j = \frac{2}{m}\sum_{i \in I} c_{ji} \in \mathbb{Z}_+$).

assignment and cost constraints (constraints [1,2] in (1)) to a tolerance of $\mathcal{O}(10^{-9})$. This ensures that the objective value of MATLAB's solutions upper bound the optimal solution for all problems.

We next compare the objective value, Z, of the MATLAB solution against the implemented methods. Here, we measured accuracy in terms of relative error in order to account for the problem size. Results are depicted in Figure 3. Although MATLAB solutions are not strictly feasible, they provide a tight upper bound on the objective function value. As shown, the greedy algorithm

Method	Feasible Solutions
Sub-gradient	200
Multiplier Adjustment	200
Greedy	200
MATLAB	14

Table 2: Feasible solutions out of 200 validation problems. B&B has a strict enforcement of the integer constraints as well as the enforcement to machine tolerance of the assignment and cost constraints. The greedy method only provides a heuristic solution and can terminate without a solution. Those cases are reflected here. MATLAB satisfies the constraints with a much lower tolerance than either of the methods implemented here.

matches the accuracy of the exact methods almost perfectly. The maximum relative error of any of the three methods is on the order of the integer tolerance of the MATLAB solver.

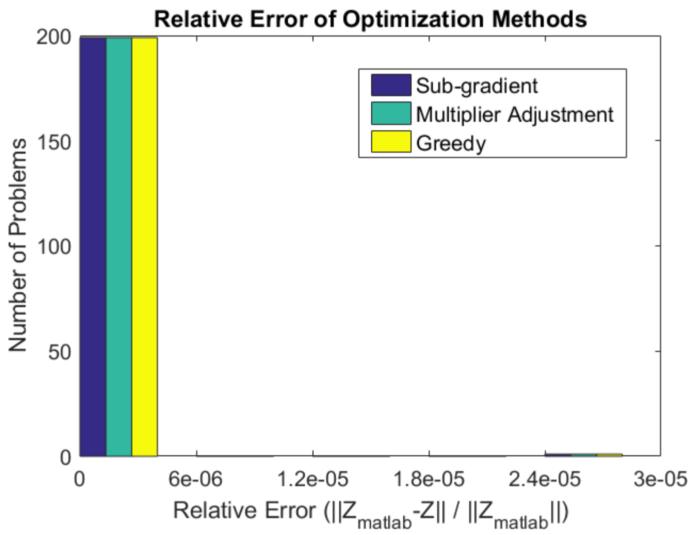


Figure 3: Histogram of the relative error of target value, Z, for greedy and B&B methods. Relative error is defined as follows: $||Z_{MATLAB} - Z_{greedy/B\&B}||/||Z_{MATLAB}||$. The accuracy of the greedy method nearly equals that of the B&B algorithm.

Beyond accuracy, we would like to compare the speed of all three methods relative to MATLAB's implementation. In order to analyze time complexity, we conducted a one-way analysis of variance (ANOVA) in which the factor is problem size. This analysis requires the assumptions that data within a given problem size is independent and identically distributed (IID), error within problem sizes is normally distributed, and variance across problem sizes is homogeneous. The first assumption is easily satisfied by our problem set-up; however, the latter require further consideration. The combinatorial nature of B&B will necessarily create a heavy-tailed effect in the data since some problems will require the algorithm to enqueue and bound many orders of magnitude more sub-problems. This effect will also result in growth of variance with problem size; however, these problems will be infrequent, and if treated as outliers, the data should easily satisfy the second and third assumptions. Using a log-log scale also helped to minimize the effects of outliers.

As shown in Figure 4, all methods grow sub-linearly with respect to the problem size. We used the greedy algorithm in the implementation of the image labeling system because it achieved the

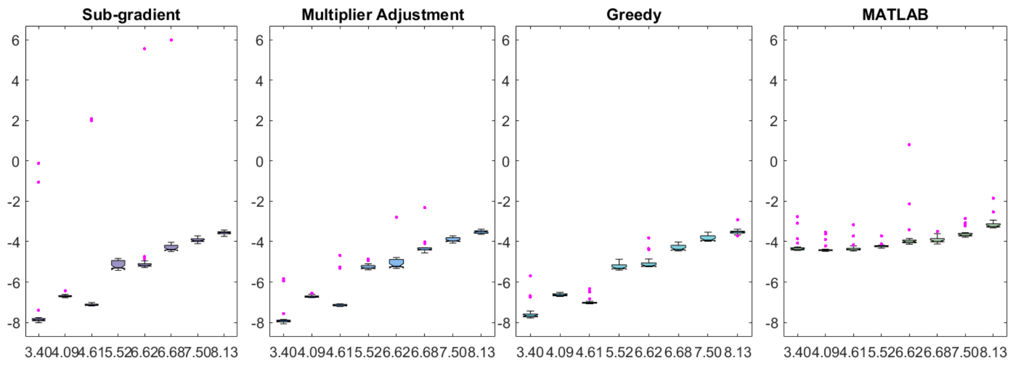


Figure 4: Analysis of variance for problem size versus computational time ($\log - \log$). The F-statistic, $F(7, 192)$, is 14.89, 343.08, 996.8, and 23.41 respectively ($p < 1.0 \times 10^{-9}$). The horizontal line in the box-plots is the median of the data. The top and bottom of the box reflect the 75th and 25th percentile respectively. The magenta dots represent outliers (greater than 2.7 standard deviations from the mean). Two medians are significantly different if their notch intervals do not overlap. The respective complexities of the methods are as follows: sub-gradient ($\mathcal{O}((m \times n)^{0.78})$), multiplier adjustment ($\mathcal{O}((m \times n)^{0.90})$), greedy ($\mathcal{O}((m \times n)^{0.87})$), and MATLAB ($\mathcal{O}((m \times n)^{0.23})$).

same performance with fewer outliers than the B&B methods. The delay in subsequent iterations would harm the performance of the system. In the interest of minimizing wall time, a fast, but sub-optimal, solution is preferred. In the event that the greedy method does not return a feasible solution, the system would default to the sub-gradient descent method.

3.2 Joint Classification

Upon the receipt of image labels from the agents, we can consider the binary decision of the agents as conditionally independent discrete random variables, $A^i_j : \{-1, 1\} \to \mathbb{R}$, and the set of decisions from m agents for a single image, $(A^i_1, \ldots, A^i_m)^t$, as a joint random variable, $\mathbf{A}^i : \{-1, 1\}^m \to \mathbb{R}$. If the true label of an image is a discrete random variable, $Y : \{-1, 1\} \to \mathbb{R}$, then we seek the decision rule, d, which maximizes $\mathbb{P}(d(\mathbf{A}^i) = y_i)$.

The obvious choice for this decision rule would be the decision which maximizes the log-likelihood [5] of the predictions of the individual classifiers:

$$d(\mathbf{a}^i) = \underset{y_i \in \{-1,1\}}{\arg\max} \sum_{j \in J} \log \mathbb{P}_{A^i_j | Y}(a^i_j | y_i)$$

3.2.1 Spectral Meta-Learner

If we define the balanced accuracy, π_j, of an agent j as

$$\pi_j = \frac{1}{2}(\psi_j + \eta_j), \tag{6}$$

where ψ_j is sensitivity, $\mathbb{P}(a^i_j = 1 | y_i = 1)$, and η_j is specificity, $\mathbb{P}(a^i_j = -1 | y_i = -1)$, then, as shown in [21], the decision rule can be written in terms of the sensitivity and specificity of each agent,

$$d(\mathbf{a}_i) = \text{sign}\left(\sum_{j=1}^{m} a_j \left(\log\left(\frac{\psi_j \eta_j}{(1 - \psi_j)(1 - \eta_j)} \right) + \log\left(\frac{\psi_j (1 - \psi_j)}{\eta_j (1 - \eta_j)} \right) \right) \right). \tag{7}$$

This form of the maximum likelihood estimate invites an expectation-maximization (EM) approach to improve the decision rule [5]. As the agents label more images, by using the decision rule labels, we can improve our estimate of the agent reliability and increase the maximum likelihood of the image label as shown in Algorithm 4.

We require a better than random initial guess for the image labels in order to initiate the EM algorithm. As shown in [21], we can infer agent sensitivity and specificity from the spectral properties of the sample covariance matrix of the agent classifications. We begin with the sample covariance matrix for $\{\mathbf{a}_i\}_{i\in I^*}$, where $I^* \subseteq I$ is a set of images, $|I^*| \geq m$, for which each agent has labeled image i:

$$\mathbf{Q} = \frac{1}{|I^*|-1}\sum_{i\in I^*}(\mathbf{a}_i - \bar{\mathbf{a}})(\mathbf{a}_i - \bar{\mathbf{a}})^T. \tag{8}$$

It can be shown that the sample covariance matrix is nearly equal to a rank one matrix, $q_{ij} \propto (\pi_i - \frac{1}{2})(\pi_j - \frac{1}{2})$, so in fact, the principal eigenvector of the sample covariance matrix has entries proportional to the balanced accuracy of the corresponding classifier. When combined with a first-order Taylor series expansion of (7) about $(\psi_j = \frac{1}{2}, \eta_j = \frac{1}{2})\ \forall j \in J$, the decision rule simplifies to

$$d(\mathbf{a}_i) = \text{sign}\left(\sum_{j=1}^{m} a_j(\pi_j - \frac{1}{2})\right), \tag{9}$$

a weighted linear combination of the decision of the individual agents by the principal eigenvector of (8). Our approach, captured in Algorithm 4, implements a variant of the SML which accepts non-fully-populated results and uses the first order approximation to not only initiate the EM algorithm but also to label images not in the set of images classified by all agents.

This procedure provides more than simply a decision rule for joint classification. The absolute value of the maximum likelihood estimate provides a measure of confidence in the image label, s_i, and the updated estimate of the agent balanced accuracy provides a measure of reliability, r_j. These values, in turn, update the assignment value, v_{ji}, in (1) for the next iteration.

3.3 System Implementation

Beyond providing a conceptual framework, our goal was to develop a system in MATLAB which achieves task parallelism across agents performing image labeling at distributed workstations while providing enough flexibility to accommodate varying ensembles of agents and assignment logic. All software reported here was developed in MATLAB R2015a and later releases.

3.3.1 Flexibility

Flexibility could include many aspects of the image labeling system, but specifically, we want to achieve flexibility in the number and type of agents as well as the logic used for image assignment. We envisioned a system which works with as few as three agents and scales to dozens of agents while also being agnostic to the type of agents connected, which should be easily facilitated since they only require the hardware module to provide a binary decision to the system. This was achieved by using an object-oriented programming paradigm (OOP).

Each remote agent runs a software module which interfaces with an abstract remote client class. The abstract class provides the image handling and communication with the central server of the system. Including a new type of agent simply requires writing a new software module which interfaces with the remote client. On the central server, a complementary object of a local agent class is instantiated for each remote agent. They provide the central server the means to send

Algorithm 4: Modified Spectral Meta-Learner

```
Data: $\{A_i\}_{i \in I}$
Result: $\{d(\mathbf{a}_i)\}_{i \in I}$,$|s_i|$,$r_j$
$\mathbf{Q} = \frac{1}{|I^*|-1} \sum_{i \in I^*} (\mathbf{a}_i - \bar{\mathbf{a}})(\mathbf{a}_i - \bar{\mathbf{a}})^T$;
$\mathbf{v} = \{\mathbf{v}_\ell | \ell = \arg\max_\ell \lambda_\ell\}$ $s.t.$ $\mathbf{Q}\mathbf{v}_\ell = \lambda_\ell \mathbf{v}_\ell$; $k = 0$;
for $i \in I^*$ do
    $s_i^k = \mathbf{v}^T \mathbf{a}_i$;
    $d^k(\mathbf{a}_i) = \text{sign}(s_i^k)$;
end
repeat
    // E-step:
    for $j \in J$ do
        $\psi_j^{k+1} = P(a_j = 1 | d^k(\mathbf{a}_i) = 1)$;
        $\eta_j^{k+1} = P(a_j = -1 | d^k(\mathbf{a}_i) = -1)$;
        $r_j^k = \frac{1}{2}(\psi_j + \eta_j)$;
    end
    // M-step:
    for $i \in I^*$ do
        $s_i^{k+1} = \left( \sum_{j=1}^{m} a_j \left( \log\left( \frac{\psi_j^{(k+1)} \eta_j^{(k+1)}}{(1-\psi_j^{(k+1)})(1-\eta_j^{(k+1)})} \right) + \log\left( \frac{\psi_j^{(k+1)}(1-\psi_j^{(k+1)})}{\eta_j^{(k+1)}(1-\eta_j^{(qk+1)})} \right) \right) \right)$;
        $d^{k+1}(\mathbf{a}_i) = \text{sign}(s_i^{k+1})$
    end
    $k = k + 1$
until $convergence$;
for $i \in I \setminus I^*$ do
    $s_i^k = \sum_{j \in J} r_j^k a_j$;
    $d^k(\mathbf{a}_i) = \text{sign}(s_i^k)$
end
```

assignments and receive labels from each remote agent. This is automatically scalable, as the system is agnostic to the number of agents in the system.

In order to achieve flexibility in the assignment logic. A separate abstract class of assignments provides the interface to the system for tracking the evolution of the system as well as generating assignments on each iteration. Again, in order to implement a new logic paradigm, it requires writing a software module which interfaces with the abstract assignment class.

A full software map is shown in Figure 5. The experiment class provides the environment in which the system runs on the central server. It has a control object which maintains results and calculates labels according to the joint classification logic. The control object has an assignment object which generates assignments and collects image labels from the remote agents. The control object also has a unique local agent object for each remote agent on the system.

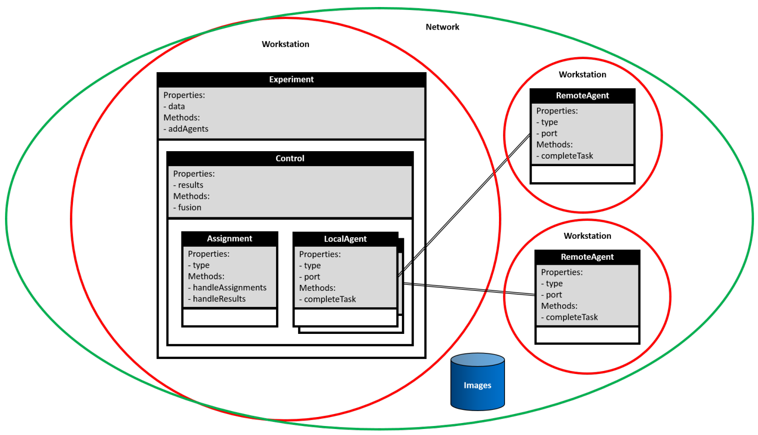


Figure 5: Software map of the image labeling system. There are five distinct classes in the system: Experiment, Control, Assignment, LocalAgent, and RemoteAgent. The RemoteAgent objects live on distributed workstations, while all other objects live on the central server. All workstations are networked with a shared image database.

3.3.2 Parallelism

Two levels of parallelism are important to the implementation of the image labeling system. There is parallelism which occurs on the central server, and that which occurs among the remote agents. Performance of the system is primarily concerned with the latter. The bulk of the wall time for system convergence will occur while the respective remote agents are labeling images. If this cannot take place in parallel, then the image labeling system cannot achieve meaningful speed-up over a serial assignment policy.

On the central server, parallelism comprises asynchronous read-write calls which facilitate run-time efficiency. Less important than sending image assignments and retrieving image labels in parallel is handling them asynchronously. The retrieving of image labels would include critical steps which would require serial processing regardless. The goal is then to allow the central server to return to idle when not actively assigning images or retrieving labels and to efficiently handle numerous run-time instructions in serial.

Much more important than parallelism on the central server is true task parallelism between the remote clients. This is facilitated by affording each agent a dedicated workstation, or at least a dedicated processor in the case of a computer vision agent, bypassing the need to share processing time. A semi-distributed memory model, where the image database is read-available to all agents, is used to alleviate the bandwidth burden on the network. This distributed workstation model with distributed memory would usually be handled with a message passing interface (MPI), but here, we are interested in multiple program multiple data programming (MPMD).

Implementing MPMD task parallelism in the image labeling system required the development of a custom MPI. A usual MPI comprises worker management, point-to-point communication, and collective calls [6]. Worker management consists of the tracking of available nodes for computation. In the image labeling system, the nodes are actual agents at distributed work stations, and the control object provides this management functionality. Point-to-point communication provides the system with a means to pass messages between and among the central server and remote clients. The local and remote agent classes provide this capability by maintaining a unique user datagram protocol socket (UDP) for each distributed agent. The UDP provides the transport layer of the internet protocol suite. It requires less overhead but affords less robustness than the more common transport control protocol [22]. Since this system only communicates internally, the required reliability has been built into the software. Collective calls provide the MPI user with a means to batch send programs and collect results from all nodes at once. The assignment object in the image labeling system assumes this responsibility. Each implementation of the abstract class has to have a function for generating and sending assignments as well as retrieving results when available.

Event-driven programming provides the final element of facilitating efficient run-time performance on the central server and achieving task parallelism between the distributed agents. By allowing the occurrence of events in the system to drive the process flow, the central server can remain idle while awaiting results from the distributed agents. The two events which drive the process in the image labeling system are the initial beginning of the experiment and the arrival of image labels to the central server. When the user starts the experiment, the process moves to assignment, where an assignment is generated and sent in serial to each agent before returning to idle. The system then remains in idle until a message arrives at a UDP socket, at which point, the assignment retrieves the labels and records them in control. The same series will occur each time a message arrives to the central server. Meanwhile, assignment tracks the status of messages which it awaits, and when all messages return, the joint classification function is called, images are classified, and a new assignment is generated and sent. The process flow is visualized in Figure 6.

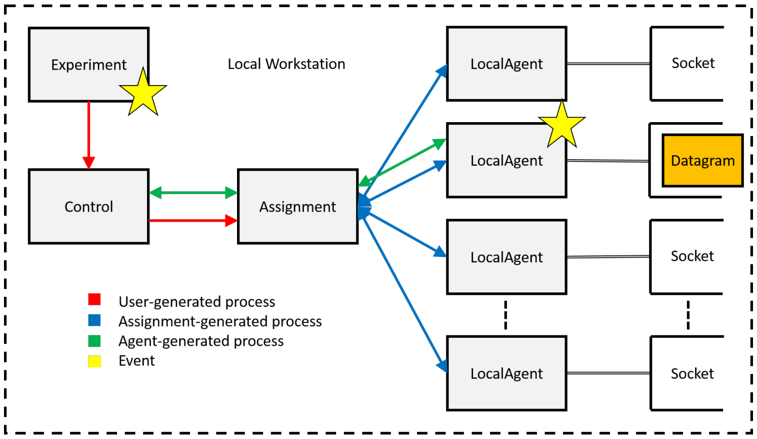


Figure 6: Process flow on central server of image labeling system. Events drive the process flow on the central server. The experimental user initiates the system. The assignment object generates assignments, sends assignments to remote clients serially through the local agent objects, and returns to idle. Upon the receipt of a datagram, assignment reads the results via the local agent module, writes the results in control, and returns to idle. System idle time is maximized through non-blocking read operations in order to facilitate the asynchronous arrival of results from the remote clients.

3.3.3 Convergence Considerations

At each iteration, the image assignment problem uses parameters estimated from the joint classification, which introduces a memoryless feedback into the system. This necessitates consideration of the stability and convergence of the system.

In order to initiate the system, prior to any inferred knowledge of the agent reliabilities or image labels, we use a batch assignment to all m agents of $m(m+1)$ images in order to adequately estimate the sample covariance matrix (8).

Considering convergence of the system, we consider the stopping conditions. We define convergence as all images labeled with sufficient confidence. Ideally, this is where the system would stop, but many things could happen to prevent this; images could be continually assigned to the same reliable agent who has already seen the image, or some images may be equally hard for all agents and never achieve sufficient confidence. In practice, there is no guarantee that all images will achieve this confidence threshold even if all agents labeled the image. This means that a confidence threshold should be some fractional value of maximum agreement of all agents. The value of this threshold will likely impact the number of agents who must see an image for it to be classified, and thus drive the wall time of the system. In order to address duplicate assignments, the value of an assignment which has previously been made is set to zero for all subsequent iterations. This is not a hard barrier, such as $v_{ji} = -\infty$ would be, but it discourages duplicate assignment unless necessary to satisfy the constraints of (1).

Stability of the system primarily depends on the assignment problem being strictly feasible. As the system evolves, and images achieve confidence threshold, there will be the same number of agents available to label fewer images; however, more of those agents will have seen some or all of the images. This will correspond to the throughput rate of the agents; computer vision agents will quickly exhaust the database, while slower human labelers will see a small fraction of it. For images which have been classified by all high throughput agents and have yet to achieve confidence threshold, it is necessary to increase the budget of low throughput agents to allow the system to make these high value assignments. Essentially, the active constraints for the system will be those of the reliable, low-throughput agents. If we can loosen these constraints, we can expedite system convergence. This is true to the point that the assignment is pseudo-infeasible, which we define as being unable to satisfy constraint [2] without zero-value assignments. This serves as an alternative stopping condition for the system. In system implementation, we incorporate both a dynamic, monotonically increasing budget for the agents, as well as an alternative stopping condition for when the assignment problem is pseudo-infeasible.

4 Methods

All simulations and the experiment ran on a Unix-based desktop computer with two Intel Xeon 2.67 GHz processors, for a total of 8 cores to support independent processes. For the simulations, agent performance was provided by simulation modules, which randomly generated both image labels and pauses for their interface with the RemoteAgent objects. This framework allowed both the simulations and the experiment to take place on a single multi-core workstation in which each agent and the central server ran on a separate dedicated instance of MATLAB.

The simulated agents generated labels randomly according to a Bernoulli distribution, $f_{A_j|Y}(a_j|y) = bern(p_j)$. The agents generated service times for each image according to an exponential distribution, $T \sim exp(\mu)$. The budget was a function of the speed of the agent, $b_j = \frac{L_k}{\mu_j}$, where L_k is the desired interval length for iteration k.

We wanted to simulate a scenario in which very little training data is available for the computer vision algorithms such that both the human and RSVP analysts possess an accuracy advantage. We encoded this expected performance into the simulation parameters, with the human agents and RSVP agents having a higher relative accuracy than the computer vision agents, while maintaining an order of magnitude difference in speed between each agent type. See Table 3 for a summary of the agent parameters. For all simulations, we measured balanced accuracy (6), wall time, the objective elapsed time while the system runs to convergence, and the number of overall assignments to reach convergence.

Assigning all images to all agents in parallel already represents a speed-up over the serial labeling of all images by all agents, but we want to specifically decrease the wall time of such a system;

Type	**Accuracy (p_j)**	**Cost (c_{ji})**	**Service Time (μ_j)**
CV	0.75	1	0.01s
RSVP	0.85	1	0.1s
Human	0.95	1	1.0s

Table 3: Properties of Simulated Agents. Human analysts provide the most accurate labels but also require the most service time for each image. Computer vision agents require the least service time but provide the least accurate labels.

therefore, we compared three assignment conditions of the proposed framework against this naive parallel implementation:

- Naive - all images assigned to all agents in parallel in a single batch.
- GAP-2 - images assigned in parallel according to (1); images classified if confidence meets or exceeds two, $s_i \geq 2$.
- GAP-3 - same as GAP-2, but with $s_i \geq 3$.
- GAP-4 - same as GAP-2, but with $s_i \geq 4$.

All methods use Algorithm 4 to combine image labels. Additionally, we considered three distinct ensembles of agents for the proposed system:

- $CV \times 6$
- $\{CV \times 2, RSVP \times 2, H \times 2\}$ (Mixed)
- $H \times 6$

4.1 Expected Results

We can determine the expected performance of the naive assignment condition analytically, which provides a true performance ceiling to which the GAP assignment conditions can be measured.

The accuracy from the joint classification using the first-order approximation of the SML is bounded from below by the accuracy of the best individual agent in the ensemble to within an additive constant (Lemma S2 (iii) of [21]). This result depends on the strict conditional independence of all classifiers and that all classifiers are better than random, which is easily satisfied by the simulated agents in this experiment.

For the naive assignment condition, the wall time for a single agent to complete classification of n images will be an Erlang random variable, $T_j \sim Erlang(n, \mu_j)$, and the system wall time will be the maximum of m independent, non-identical Erlang distributions, $T = \max_{j \in J} T_j$. We can numerically evaluate the probability density function,

$$\begin{aligned} f_T(t) &= \frac{\partial}{\partial t} \mathbb{P}(\max_{j \in J} T_j \leq t) \\ &= \frac{\partial}{\partial t} \mathbb{P}(T_1 \leq t, \ldots, T_m \leq t) \\ &= \frac{\partial}{\partial t} \mathbb{P}(T_1 \leq t) \cdots \mathbb{P}(T_m \leq t) \\ &= \frac{\partial}{\partial t} \prod_{j \in J} F_{T_j}(t) \end{aligned}$$

$$= \left(\prod_{j \in J} F_{T_j}(t) \right) \sum_{j \in J} \frac{f_{T_j}(t)}{F_{T_j}(t)}$$

to calculate the mean, $\mu_T = \mathbb{E}(T)$, and standard deviation, $\sigma_T = \sqrt[2]{\mathbb{E}((\mu_T - T)^2)}$, of the system wall time.

4.2 Simulation 1

We compared the performance of a mixed ensemble over all four assignment conditions and collected results from 30 trials of six remote agents classifying 200 images for each assignment condition.

4.3 Simulation 2

We compared the performance of three distinct agent ensembles using the GAP-2 assignment condition: $CV \times 6$, Mixed, and $H \times 6$. These results provide context to any speed-up in the results of Simulation 1 by comparing the mixed agent ensemble against a fully automated implementation and a fully human ensemble. Again, 30 trials were collected for six remote agents classifying 200 images.

4.4 Experiment

In the final experiment, we applied the proposed framework with actual agents. We compare the performance of the $CV \times 6$ agent ensemble against another mixed ensemble ($\{CV \times 5, H\}$) using the GAP-2 assignment strategy. For our image database, we used a subset of the Caltech101 database [7]. We used the background category of images as the negative case and the brain category of images as the positive case. We tested on 200 pre-selected images, 150 background and 50 brain. The computer vision algorithms used a pre-trained network [3] from MatConvNet [24] to extract features from images, and the MATLAB internal support vector machine classifier with all default settings to classify the features. The computer vision algorithms were trained on 20 randomly selected remaining images from each of the background and brain categories, for a total of 40 training images, uniformly sampled from both classes. This produced very accurate computer vision algorithms (population balanced accuracy of 0.922 ± 0.020). The human labels were provided by me through a single-image display with a two-button interface.

5 Results

5.1 Analytical Results

The analytical results for all ensemble conditions under the naive assignment condition are reported in Table 4. The mixed agent ensemble matches the lower bound of balanced accuracy of the human ensemble while benefiting from a lower expected wall time. With the automated ensemble, a $95\times$ and $99\times$ fold speed-up can be expected versus the mixed agent and human ensembles respectively, but this speed-up incurs a 21% decrease in the lower bound of balanced accuracy.

5.2 Simulation 1

All four methods exceeded the analytical lower bound of the mixed agent ensemble. The results are summarized in Table 5. We first performed a one-way analysis of variance (ANOVA) which

Agent Ensemble	**Accuracy (π_j)**	**Wall Time ($\mu_T \pm \sigma_T$)**
$CV \times 6$	0.75	2.2 ± 0.1s
Mixed	0.95	208.0 ± 12.0s
$H \times 6$	0.95	218.3 ± 9.7s

Table 4: Analytical results of naive assignment condition.

Condition	**Balanced Accuracy**	**Wall Time**	**Images Assigned**
Naive	0.988 ± 0.011	204.1 ± 7.9	1200
GAP-2	0.974 ± 0.014	124.1 ± 19.3	879.9 ± 16.3
GAP-3	0.975 ± 0.011	147.9 ± 21.8	983.1 ± 15.1
GAP-4	0.978 ± 0.011	204.4 ± 12.3	1047.6 ± 6.4

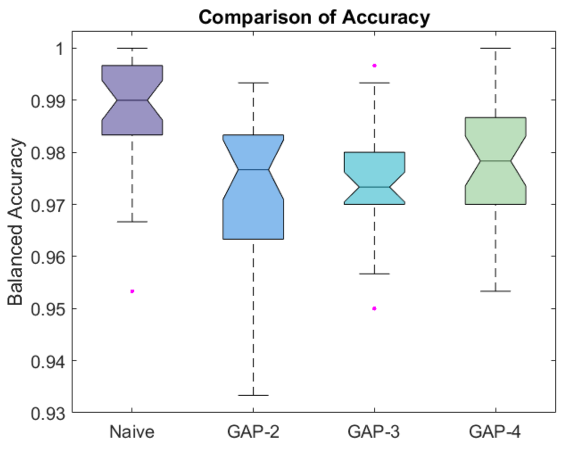
Table 5: Results of Simulation 1. The mean and standard deviation are reported for the balanced accuracy, wall time, and number of images assigned for the varying assignment conditions with the mixed ensemble.

reveals significant differences among the assignment strategies (see Figure 7(a)). For the three GAP assignment conditions, there is not a significant difference in the means of the data at the $p < 0.01$ level, but there is a significant difference between the GAP conditions and the control in a multiple comparisons test (GAP-2: $p = 7.7 \times 10^{-5}$, GAP-3: $p = 2.4 \times 10^{-4}$, GAP-4: $p = 3.8 \times 10^{-3}$).

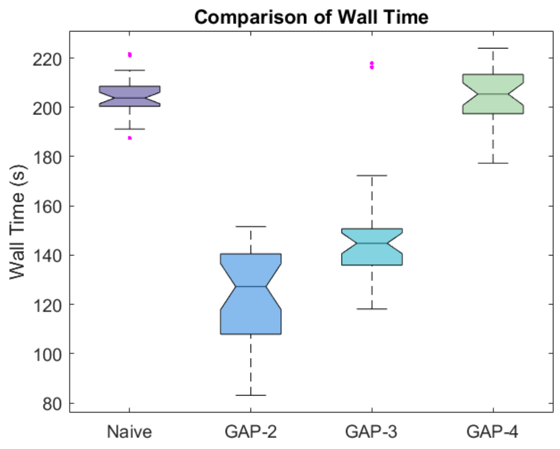
In the case of wall time, much more noticeable differences arise. An ANOVA reveals significantly different mean wall times between the assignment conditions (see Figure 7(b)). GAP-2 and GAP-3 achieve significantly lower run-times than the naive assignment condition under a multiple comparisons test (GAP-2: $p = 3.8 \times 10^{-9}$, GAP-3: $p = 3.8 \times 10^{-9}$). The mean of the GAP-2 condition achieves a 1.6× speed-up over the mean of the naive condition, while the GAP-3 achieves a 1.4× speed-up.

Again, an ANOVA reveals significant differences between the number of assignments ($F(2, 87) = 1205.8, p < 1.0 \times 10^{-9}$). Under a multiple comparisons test, the number of overall assignments for all assignment conditions is significantly different than that for all other assignment conditions at the $p < 1.0 \times 10^{-9}$ level.

The mixed agent system clearly achieved superior wall time under the GAP-2 and GAP-3 assignment conditions versus the naive assignment condition, but the results of the accuracy were less clear. Although the naive condition achieved significantly better balanced accuracy than any of the GAP conditions, we are more specifically testing the hypothesis that the GAP conditions achieved or exceeded the analytical lower bound of the naive assignment condition. In this light, the results are more favorable. Under a one sample t-test, it is almost certain that the accuracy of the mixed ensemble GAP assignment conditions achieved or exceeded the analytical lower bound of the mixed ensemble naive assignment (p-values of 8.5×10^{-11}, 7.7×10^{-14}, and 1.3×10^{-14} for the GAP-2, GAP-3, and GAP-4 conditions respectively). Under this more relaxed hypothesis, the GAP-2 and GAP-3 conditions achieved the accuracy performance of the naive assignment condition while significantly decreasing the wall time required to do so.



(a) Balanced Accuracy



(b) Wall Time

Figure 7: ANOVA of the performance of heterogeneous agent ensembles across assignment conditions reveals significance in both the balanced accuracy ($F(3, 116) = 8.8$, $p = 2.6 \times 10^{-5}$) and wall time ($F(3, 116) = 186.5$, $p < 1.0 \times 10^{-9}$). Results reported as in Figure 4.

5.2.1 Increased Confidence

It should be expected that increasing the confidence threshold of the proposed system should have a measurable impact on the confidence scores, s_i, of the images. As shown in Figure 8, the All condition achieves a higher aggregate confidence score, but among the GAP conditions, there is direct correspondence between increased threshold and increased expected image confidence.

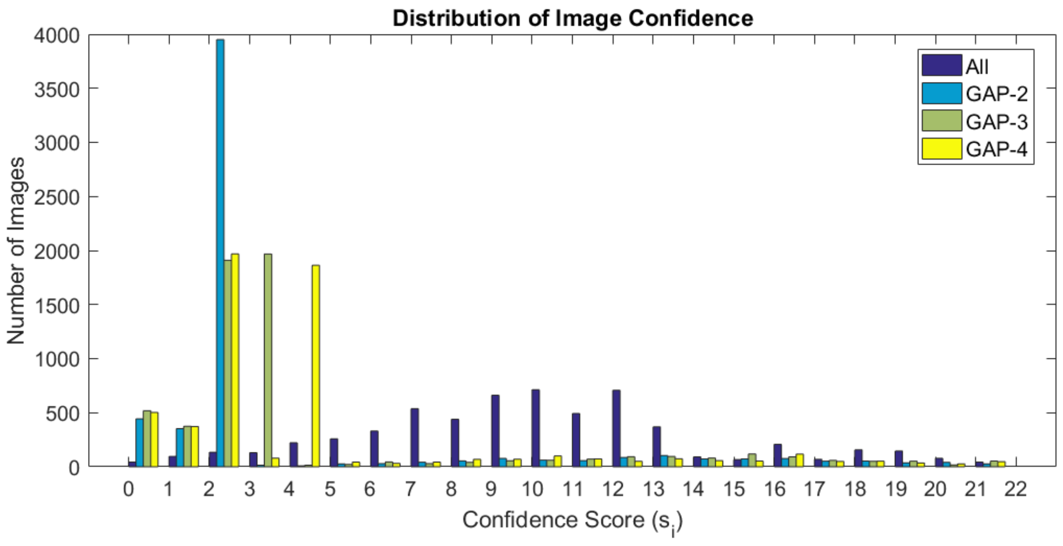


Figure 8: Comparison of the distribution of confidence scores across the four conditions. The All condition achieves a much higher mean confidence (10.3) than any of the GAP conditions (GAP-2: 5.1, GAP-3: 5.3, and GAP-4: 5.6).

This also provides a visualization of how the proposed framework achieves the decrease in wall time. The All condition benefits from the additional assignments and classifications to push more images to an increased likelihood, but the GAP conditions use only enough assignments of each image to achieve the minimal threshold. This is reflected in the number of image assignments reported in Tables 5 and 6.

5.2.2 Evolution of Intervals

Due to the dynamic setting of the budget constraint, there is a temporal evolution of the classification intervals. At first, the computer vision algorithms classify as many images as possible, but as they exhaust the image database, the interval increases to allow the lower throughput labelers to classify more images. This is particularly pronounced in the GAP-4 condition of Figure 9, where we see significant divergence at iteration 5 in the mean interval length between the three GAP conditions.

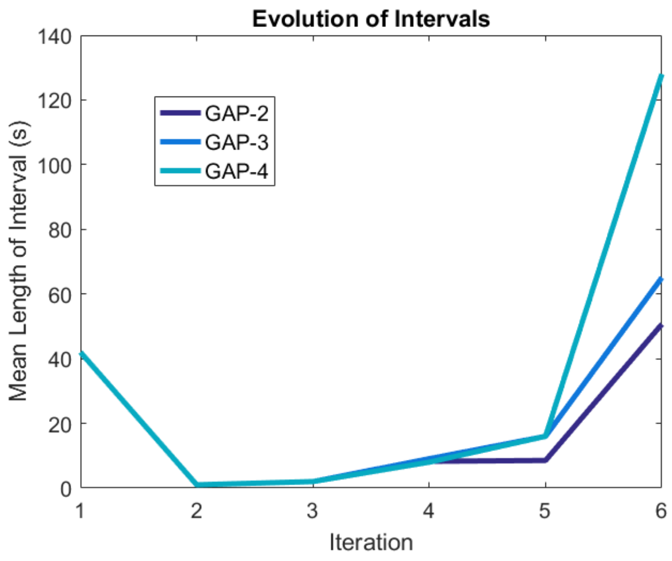


Figure 9: Comparison of the mean length of intervals in seconds among the GAP conditions. The duration of intervals mirror each other until iteration 5, where there is a significant deviation as the higher threshold method increases the interval to allow the lower throughput labelers to classify more images.

5.3 Simulation 2

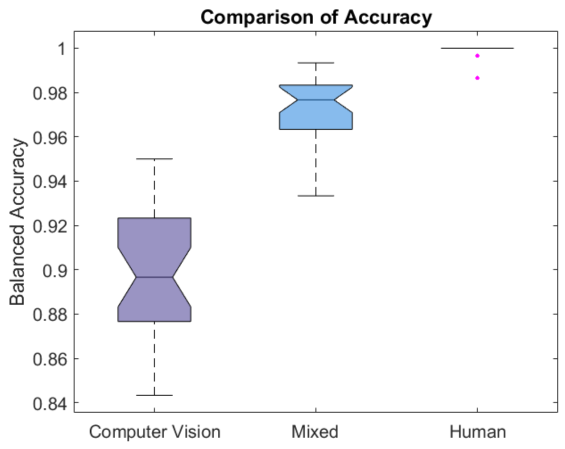
Not surprisingly, the ensemble with computer vision agents achieved the minimum wall time and also the minimum accuracy, and the ensemble of human agents achieved the maximum accuracy and the maximum wall time. The effect of varying the agent ensemble while using the GAP-2 framework mirrored the analytical results of the naive assignment condition. Results are summarized in Table 6.

Significant differences in balanced accuracy are confirmed under a one-way ANOVA (see Figure 10(a)). Specific significant differences between the mixed ensemble and the automated ($p < 1.0 \times 10^{-9}$) and human (2.2×10^{-6}) ensembles are confirmed under a multiple comparisons test. Significant differences also arose in wall time (see Figure 10(b)), and under a multiple comparisons test, the automated and human ensembles were significantly different than the mixed ensemble, $p < 1.0 \times 10^{-9}$ and $p < 1.0 \times 10^{-9}$ respectively. Similar results were found in the overall number of assignments to reach system convergence between the three ensembles ($F(2, 87) = 1000.9, p <$

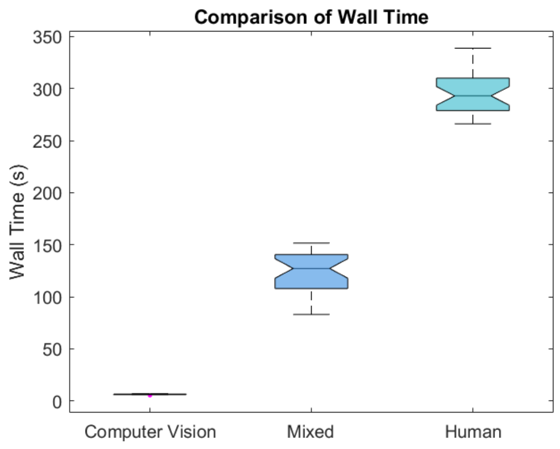
Ensemble	**Balanced Accuracy**	**Wall Time**	**Images Assigned**
$CV \times 6$	0.898 ± 0.030	6.3 ± 0.3s	913.8 ± 13.8
Mixed	0.974 ± 0.014	124.1 ± 19.3s	879.9 ± 16.3
$H \times 6$	0.999 ± 0.003	294.2 ± 18.3s	770.1 ± 7.2

Table 6: Results of Simulation 2. The mean and standard deviation are reported for balanced accuracy, wall time, and the number of images assigned for varying agent ensemble conditions using the GAP-2 assignment strategy.

1.0×10^{-9}). Under a multiple comparisons test, the mixed ensemble requires a significantly different number of assignments than either the fully-automated ensemble ($p < 1.0 \times 10^{-9}$) or the human ensemble ($p < 1.0 \times 10^{-9}$).



(a) Balanced Accuracy



(b) Wall Time

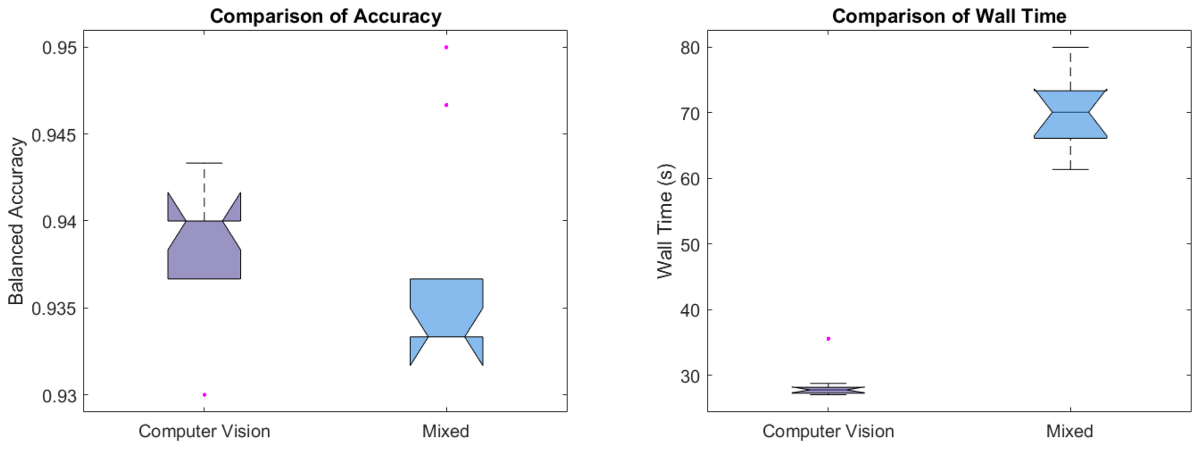
Figure 10: ANOVA of the performance of GAP-2 assignment condition across agent ensembles reveals significance in both balanced accuracy ($F(2, 87) = 255.47$, $p < 1.0 \times 10^{-9}$) and wall time ($F(2, 87) = 2667.44$, $p < 1.0 \times 10^{-9}$). Results reported as in Figure 4.

5.4 Experiment

Ensemble	**Balanced Accuracy**	**Wall Time**	**Images Assigned**
$CV \times 6$	0.938 ± 0.005	28.5 ± 2.5s	775.4 ± 8.0
Mixed	0.937 ± 0.006	70.2 ± 6.0s	774.7 ± 4.0

Table 7: Results of Experiment. The mean and standard deviation are reported for balanced accuracy, wall time, and the number of images assigned for varying agent ensemble conditions using the GAP-2 assignment strategy.

The results of the experiment do not reveal much difference in the behavior of the computer vision ensemble and mixed ensemble besides a 246% increase in wall time from adding a human analyst. We do not observe a significant difference in balanced accuracy or the number of images assigned at the $p < 0.01$ level.



(a) Balanced Accuracy (b) Wall Time

Figure 11: ANOVA of the performance of GAP-2 assignment condition across agent ensembles in the experiment. We do not observe a significant difference in balanced accuracy between the two methods ($F(1, 18) = 0.17$, $p = 0.68$), but an analysis of wall time does reveal a significant difference ($F(1, 18) = 405.03$, $p < 1.0 \times 10^{-9}$). Results reported as in Figure 4.

6 Discussion

We presented an image triage system which leverages the collaboration of heterogeneous agents, and we demonstrated the benefit of such a system versus a naive parallel implementation or a similar homogeneous ensemble of agents in simulation. Three types of agents were simulated, representing varying levels of accuracy and throughput. The system dynamically inferred the performance of these agents using the SML and incorporated that information into subsequent assignments using the GAP.

Even in the naive parallel implementation, the performance of the mixed agent ensemble provides a superior lower bound in balanced accuracy to that of the automated ensemble. Additionally, the mixed ensemble provides a decrease in the expected wall time of the system. This substantial increase in accuracy requires a sacrifice of a similar scale in wall time. This trade-off underlies the challenge of optimizing such a system, but the proposed image triage system attempts to mitigate the time cost of this trade-off through an intelligent assignment framework.

In Simulation 1, we observed the same mixed ensemble achieve similar accuracy in significantly less time under the GAP assignment conditions. For the case of the GAP-2 assignment condition, the system exceeded the guaranteed accuracy of the mixed ensemble naive assignment condition while also affording a 1.6× speed-up over the mixed ensemble naive assignment condition. It apparently accomplished this time savings by making 27% fewer overall image assignments.

The proposed framework minimizes the number of assignments in the system to achieve a desired confidence. This goal is similar in nature to the CrowdSynth system proposed in [14], where the control logic attempts to calculate the marginal value of recruiting another agent for labeling the image. Here, that decision is encoded in the GAP problem and made simultaneously over all images. Interestingly, we observed a speed-up from this approach in the mixed ensemble, but the automated ensemble and human ensemble required more time in the GAP assignment conditions than that expected from the naive assignment condition. The time cost of this additional decision process is apparently detrimental in the case of a homogeneous ensemble, which corroborates the work of Karger, et al. in [15].

In the experiment with actual agents, we did not pose a problem of sufficient difficulty to prove our findings. In this scenario, the accuracy differential between the computer vision agents and the human analyst does not provide a sufficient advantage to the value of image assignments to the low-throughput agent to justify the time cost. It is likely that in a better real-world implementation, the results of incorporating a human into the system will be even more significant than in the simulations. In particular, all agents here provided truly conditionally independent labels in the simulations; however, this quality could be hard to realize in practice. Even different models trained on the same data will introduce conditional dependence to the system. In application, heterogeneous agents may provide the only means of introducing the necessary independence into the ensemble.

7 Conclusion

In simulation, the proposed image triage system achieved human-level accuracy while minimizing the time required to do so. These results introduce a framework for realizing the advantage of the collaboration of humans and intelligent systems on a common task. The introduction of a human to a less accurate automated image triage system can instantly increase the expected accuracy of the system to the performance ceiling, and an intelligent assignment policy can minimize time cost incurred to do so. Future work will confirm these findings in a real-world application of the image triage system introduced here with human analysts, RSVP analysts, and computer vision agents.

A Multiplier Adjustment Method

Algorithm 5: Multiplier Adjustment Method

```
Data: $\lambda_i = \max_2 v_{ji}$, $x_{ji} = 0$, $\forall\ j \in J, i \in I$
Result: $\mathbf{x}, Z$
for $j \in J$ do
    % Assign tasks with value greater than the multiplier according to the knapsack problem
    $I_j^+ = \{i \in I | v_{ji} - \lambda_i > 0\}$;
    $[x_{j,I_j^+}, Z_j] = knapsack(\{v_{ji} - \lambda_i\}_{i \in I_j^+}, \{c_{ji}\}_{i \in I_j^+}, b_j)$;
end
while do
    % Assign unassigned tasks with a value equal to the multiplier to capable agents
    $\bar{I} = \{i \in I | \sum_{j \in J} x_{ji} = 0\}$; $I_j^0 = \{i \in \bar{I} | v_{ji} = \lambda_i\}$; $J_i^0 = \{j \in J | i \in I_j^0\}$; $\bar{b}_j = b_j - \sum_{i \in I} c_{ji} x_{ji}$;
    $\mathbf{x} = \arg\max_{\mathbf{x}} \sum_{j \in J_i^0} \sum_{i \in I_j^0} v_{ji} x_{ji}$, such that $\sum_{j \in J_i^0} x_{ji} \leq 1$, $i \in \bar{I}$, $\sum_{i \in I_j^0} c_{ji} x_{ji} \leq \bar{b}_j$, $j \in J$,
    $x_{ji} \in \{0,1\}$, $c_{ji}, b_{ji} \in \mathbb{Z}_+$, $j \in J_i^0$, $i \in I_j^0$;
    if $\boldsymbol{x}$ is feasible then
        return; % Assignment is optimal in GAP
    end
    for $i = \{i \in I | \sum_{j \in J} x_{ji} = 0\}$ do
        % For unassigned tasks, find the least decrease in the multiplier to assign task
        for $j \in J$ do
            $\delta_{ji} = Z_j(u) - v_{ji} + \lambda_i - knapsack(\{v_{jl} - \lambda_l\}_{l \in I | l \neq i}, \{c_{jl}\}_{l \in I | l \neq i}, b_j)$;
        end
    end
    $I^0 = \{i \in I | \sum_{j \in J} x_{ji} = 0, \min_2(\delta_{1i}, \ldots, \delta_{mi}) > 0\}$;
    if $I^0 = \emptyset$ then
        return; % No more optimal assignment exists
    else
        % Attempt to assign tasks with minimal decrease in multiplier required for the task
        to be assigned. If assignment is more optimal, then continue, else select another task.
        for $i^* \in I^0$ do
            $j^* = \arg\min(\delta_{1i^*}, \ldots, \delta_{mi^*})$; $x_{j^*i^*} = 1$;
            $[x_{j^*, i \in I | i \neq i^*}, Z_{j^*}] = knapsack(\{v_{j^*i} - \lambda_i\}_{i \in I | i \neq i^*}, \{c_{j^*i}\}_{i \in I | i \neq i^*}, b_{j^*})$;
            $I^0 = I^0 \setminus \{i^*\}$;
            if $\sum_{i \in I} x_{ji} \leq 1\ \forall j \in J$ then
                return to start of while loop;
            else
                revert $\mathbf{x}$ and try another $i^*$;
            end
        end
    end
end
```

References

[1] Nima Bigdely-Shamlo, Andrey Vankov, Rey R Ramirez, and Scott Makeig. Brain activity-based image classification from rapid serial visual presentation. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 16(5):432–441, 2008.

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

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[4] Jens Clausen. Branch and bound algorithms-principles and examples, 1999.

[5] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[6] Victor Eijkhout. *Introduction to High Performance Scientific Computing*. Lulu. com, 2014.

[7] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Int. Conf. Comput. Vis. Patt. Recogn. Workshop on Generative-Model Based Vision*, pages 59–70, 2004.

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

[9] 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.

[10] Chien-Ju Ho, Shahin Jabbari, and Jennifer W Vaughan. Adaptive task assignment for crowd-sourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 534–542, 2013.

[11] Qingyang Hu, Qinming He, Hao Huang, Kevin Chiew, and Zhenguang Liu. A formalized framework for incorporating expert labels in crowdsourcing environment. *Journal of Intelligent Information Systems*, pages 1–23, July 2015.

[12] Panagiotis G. Ipeirotis, Foster Provost, Victor S. Sheng, and Jing Wang. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*, 28(2):402–441, March 2013.

[13] Ajay J Joshi, Fatih Porikli, and Nikolaos P Papanikolopoulos. Scalable active learning for multiclass image classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2259–2273, 2012.

[14] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[15] David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-Optimal Task Allocation for Reliable Crowdsourcing Systems. *Operations Research*, 62(1):1–24, February 2014.

[16] 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.

[17] Jan V Leeuwen, AR Meyer, M Nival, et al. *Handbook of theoretical computer science: algorithms and complexity*. MIT Press, 1990.

[18] 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.

[19] Angelia Nedić. *Subgradient methods for convex minimization.* 2002.

[20] Ralph Otten. Lecture 13: The knapsack problem.

[21] Fabio Parisi, Francesco Strino, Boaz Nadler, and Yuval Kluger. Ranking and combining multiple predictors without labeled data. *Proceedings of the National Academy of Sciences*, 111(4):1253–1258, January 2014.

[22] J Postel. Request For Comment 768. 1980.

[23] P. Sajda, E. Pohlmeyer, Jun Wang, L.C. Parra, C. Christoforou, J. Dmochowski, B. Hanna, C. Bahlmann, M.K. Singh, and Shih-Fu Chang. In a Blink of an Eye and a Switch of a Transistor: Cortically Coupled Computer Vision. *Proceedings of the IEEE*, 98(3):462–478, March 2010.

[24] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. 2015.

[25] Yan Yan, Rmer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327, October 2013.