Assessing Optimal Assignment under Uncertainty: An Interval-based Algorithm

Lantao Liu and Dylan A. Shell
Department of Computer Science & Engineering
Texas A&M University
College Station, Texas, USA
Email: {lantao,dshell}@cse.tamu.edu

Abstract

We consider the problem of multi-robot task-allocation when robots have to deal with uncertain utility estimates. Typically an allocation is performed to maximize expected utility; we consider a means for measuring the robustness of a given optimal allocation when robots have some measure of the uncertainty (*e.g.*, a probability distribution, or moments of such distributions). We introduce the Interval Hungarian algorithm, a new algorithm that extends the classic Kuhn-Munkres Hungarian algorithm to compute the maximum interval of deviation, for each entry in the assignment matrix, which will retain the same optimal assignment. The algorithm has a worst-case time complexity of $O(n^4)$; we also introduce a parallel variant with $O(n^3)$ running time, which is able to exploit the concurrent computing capabilities of distributed multi-robot systems. This provides an efficient measurement of the tolerance of the allocation to the uncertainties and dynamics, for both a specific interval and a set of interrelated intervals. We conduct experiments both in simulation and with physical robots to validate the approach and to gain insight into the effect of location uncertainty on allocations for multi-robot multi-target navigation tasks.

1 Introduction

Task-allocation mechanisms are among the most successful non-domain-specific means for coordinating multiple robots. These involve treating the work to be performed as a set of tasks and the robots themselves as workers to be assigned to particular tasks. If each robot can estimate the efficacy of its performance for each task, algorithms can optimize the allocation of robots to tasks, or vice versa (which is equivalent), in order to maximize expected collective performance. The complexity of the allocation problem depends on the particular capabilities required to achieve the task, the capabilities of the robots, and whether the allocation must consider temporal scheduling aspects [Gerkey and Matarić, 2004]. Generally, the complexity of the particular allocation problem depends on the degree to which each of the robots and each of the tasks can be considered as independent of one another. However, a certain degree of independence is necessary for the task-allocation approach to be considered appropriate at all.

This paper considers an archetype multi-robot task-allocation problem which involves performing an instantaneous assignment of robots to tasks. Each robot is capable of performing exactly one task at a given time and each task requires only one robot. In the taxonomic characterization in Gerkey and Matarić [2004], this is the single-task robots, single-robot tasks problem instance. This reduces to the well-studied Optimal Assignment Problem (OAP) for which the Kuhn-Munkres Hungarian algorithm, which was first proposed by H. W. Kuhn [Kuhn, 1955] and improved by J. Munkres [Munkres, 1957], is a solution.

However, outstanding issues remain for multi-robot task-allocation. The Hungarian algorithm maximizes the utility for the team because it is provided with an estimate of each robot's expected utility. Calculating this estimate is costly because every robot must provide estimates for each task and the optimality of the resultant allocation is only meaningful when these estimates are accurate. Dynamic environments, dynamic tasks, or changes in robot state typically mean that utility estimates can become outdated. Furthermore, the robots have to deal with uncertainty about the state of the world in constructing these estimates. Even if the robots maintain a representation of this uncertainty (*e.g.,* a distribution over potential states) the expected utility is only the first moment of the utility distribution given a particular robot-task assignment pair. We identify the following important questions:

1. How much effort should the robots invest in constructing each utility estimate? For n robots and n tasks, n^2 estimates are provided, but only n elements make up the optimum assignment; not all utility estimates need to be known with equal fidelity.

2. Once an allocation is computed, how stable is the allocation with respect to changes in the matrix of utility estimates? Changes in the matrix can arise either from improved estimates of task performance, or tasks with some inherent dynamics.

3. If these utility estimates arise from an underlying probability distribution, what is the likelihood that the assignment is sub-optimal?

This paper makes the following contributions toward addressing the problem of multi-robot task allocation with uncertain utility values:

- This paper introduces the concept intervals of utility estimates, identifies several properties of these intervals, and provides proofs of these properties.

- We present a new algorithm, the Interval Hungarian Algorithm, which is ideally suited to multi-robot systems, although broadly applicable to any OAP where the matrix of utility estimates is subject to uncertainty.

- This paper introduces an efficient method for quantifying the effects of uncertainty on an allocation. This includes analysis for instances with a single specific estimate as well as multiple interrelated estimates.

- We analyze the impact of uncertainty through concrete examples, using standard localization methods to produce real utility distributions with physical robots in addition to simulation.

2 Related Work

It is worthwhile to draw a distinction between multi-robot coordination strategies that employ static notions of expected utility and those which model the task performance itself in greater detail. The latter schemes use a rich model of agents and tasks in order to construct a probabilistic model. For example, stochastic games/decentralized-MDPs [Bernstein *et al.*, 2000], factored-MDPs,[Guestrin *et al.*, 2002], POMDPs [Roth *et al.*, 2006], permit one to explicitly address the question of when to perform particular actions (movement, sensing, communication) in order to reduce uncertainty if doing so is beneficial for the performance of tasks. However, these problems do not admit polynomial-time solutions and often factorization or independence assumptions are introduced in order to make the problem tractable. Further constraints may arise from a distributed solution (*e.g.,* game theoretic models which design individual pay-off matrices so that locally greedy agents maximize global pay-off) and require approximation (*e.g.*, potential games, or a similar treatment) or task-structure assumptions.

Task assignment methods that employ static expected utilities instead abstract away details of the task performance and structure. They require that each robot assess their expected performance on each task under consideration. These approaches depend on an independence assumption because a linear function (like the sum) of the group's utilities is optimized. Other effects, like interference or inter-robot synergy, can only be captured in the way they effect particular utilities in the expectation. Centralized and distributed algorithms for performing allocations have been developed, including greedy allocations [Parker, 1998], optimization techniques [Gerkey and Matarić, 2004; Atay and Bayazit, 2006], and auction [Bertsekas, 1990; Berhault *et al.*, 2003] and market-based approaches [Dias and Stentz, 2001; Dias *et al.*, 2006]. It is within this framework that the present study falls. It uses little information about the domain-specific aspects that lead to the structure of the coordination problem, or even the source of the uncertainty. Despite the large body of literature concerned with efficient solution of the distributed assignment problem, comparatively little work considers the effect of uncertainty. Aspects like interrelated utilities (or in the present study, more generally, interrelated uncertainty in the utilities) are not explicitly captured. If the effect of these higher-order interactions on utility values can be calculated, then the presented algorithm can still be of use.

The Interval Hungarian algorithm introduced in this paper is similar to the broader strategy of sensitivity analysis [Dinkelbach, 1969] used to assess robustness to error of linear (or non-linear) programs. The appendix describes the assignment problem from this perspective and outlines the aspects that are unique due to our treatment of the sensitivity analysis within the bipartite graph matching framework.

3 Background: The Hungarian Algorithm

The Hungarian algorithm treats the optimization OAP as a combinatorial problem in order to efficiently solve an $n \times n$ task assignment problem in $O(n^3)$ time. The utility estimates become edge weights in a complete bipartite graph in which robots and tasks each are represented by the vertices. The Hungarian algorithm (Algorithm 3.1) searches for a perfect matching in a sub-graph of the complete bipartite graph, where the perfect matching is exactly the optimal assignment solution. In step 4 the search process either increases the matching size, or grows the so-called *equality graph* in which the matching must reside.[‡]

Algorithm 3.1 The Hungarian Algorithm

```
Input:
    An $n \times n$ utility matrix represented as the equivalent complete weighted bipartite graph $G = (X, Y, E)$, where $|X| = |Y| = n$.
Output:
    A perfect matching, $M$.
1: Generate an initial labeling $l$ and matching $M$ in $G_e$.
2: If $M$ is perfect, terminate. Otherwise, pick a random exposed vertex $u \in X$.
   Set $S = \{u\}$, $T = \varnothing$.
3: If $N(S) = T$, update labels:
   $\delta = min_{x \in S, y \in Y-T}\{l(x) + l(y) - w(x,y)\}$
   $l'(v) = \begin{cases} l(v) - \delta & if\ v \in S \\ l(v) + \delta & if\ v \in T \\ l(v) & otherwise \end{cases}$
4: If $N(S) \neq T$, pick $y \in N(S) - T$.
   (a) If $y$ exposed, $u \rightarrow y$ is augmenting path. Augment $M$ and go to step 2.
   (b) If $y$ matched, say to $z$, extend Hungarian tree: $S = S \bigcup \{z\}$, $T = T \bigcup \{y\}$,
   and go to step 3.
```

$*$ Definitions:

- Equality graph $G_e = \{e(x,y) : l(x) + l(y) = w(x,y)$;
- Neighbor $N(u)$ of vertex $u \in X$: $N(u) = \{v : e(u,v) \in G_e\}$.

[‡]Graph and matching definitions and notation are adopted from Lovász and Plummer [1986].

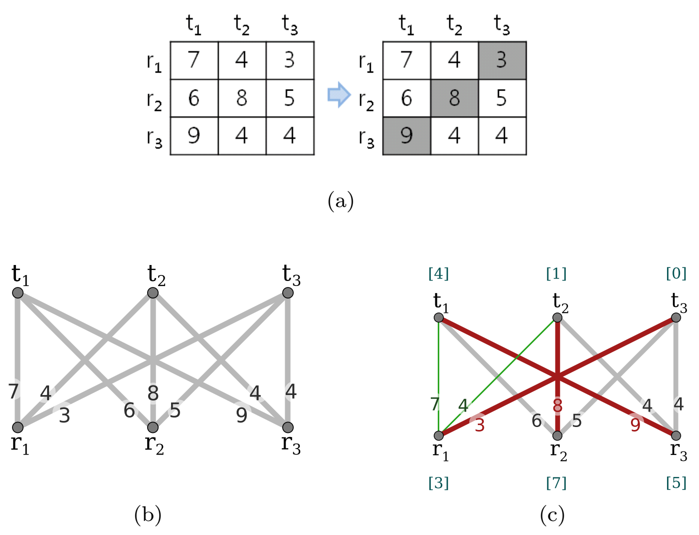


Figure 1: The Hungarian algorithm solves the OAP using the input represented as a complete bipartite graph. (a) An assignment matrix and solution; (b) A complete bipartite graph; (c) The perfect matching for assignment solution. The state of the graph after an assignment, we term the *resultant bipartite graph*. (Viewed in gray scale the so called green edges are thiner, red edges are the darker bold edges, gray edges are the lighter bold edges.),

Figure 1 shows an example assignment problem and the corresponding perfect matching in the form of the associated bipartite graph. In Figure 1(a), the task assignment problem is described as an assignment matrix (which, in general, need not be square). Element u_{ij} represents the utility that is estimated to result from assigning robot r_i to perform task t_j. The algorithm generates the maximal allocation, shown as shaded cells in the matrix. Figures 1(b) and 1(c) show the same information in the equivalent bipartite graph form. We illustrate the different way edges are used in the calculation of an allocation with color: red edges comprise the matching, green edges are unmatched whilst also being in the equality graph G_e, and gray edges are unmatched but do not appear in the equality graph. Edges within G_e are termed admissible (both red and green edges) and will have weights that satisfy $w_{ij} = l(r_i) + l(t_j)$. Bracketed integers above and below each vertex represent the labeling value $l(\cdot)$. Edge set M and scalar m represent a specific perfect matching solution and the corresponding optimum value, respectively.

Since the development of the Hungarian algorithm many variations have been proposed; see Pentico [2007] for a recent survey. Two relate directly to this paper: Toroslu and Üçoluk [2007] provide an incremental Hungarian Method, which is an $O(n^2)$ technique for inserting a pair of new vertices in the bipartite graph that has resulted from a previous assignment. An extension to this (in Mills-Tettey *et al.* [2007]) also permits deletions, which enables one to solve assignment problems with k utility changes in $O(kn^2)$ time. The current work computes a description of an assignment problem (in worst case of $O(n^4)$ sequential computation) which permits subsequent $O(1)$ queries of whether or not the assignment has changed. When assignments are being recomputed frequently with small absolute changes in value this can be a considerable saving. This happens, for example, to estimated distances during replanning as a robot moves.

4 Interval Hungarian Algorithm

The Interval Hungarian Algorithm is developed in this section. The primary purpose of the algorithm is to efficiently compute both the optimal assignment and additional information permitting one to assess the robustness of the assignment to perturbation of the input. For each utility value, we compute the interval in which the utility may be (independently) altered before the optimality of the computed assignment is violated. Thus, given an input matrix of utilities, the algorithm characterizes a set of inputs which yield the same output. The set of inputs is characterized by *intervals* which are computed on the basis of the three different categories (or three colors) of edge described in the previous section.

Before presenting the algorithm, we will describe the properties of matched and unmatched edges respectively. Matched edges are the most straightforward an permit the reader to get a feel for what is meant by the "interval analysis." The properties of the classes of edge lead to the main algorithm description, and the final subsection presenting the computational complexity of the algorithm.

4.1 Interval Analysis for Matched Edges

Consider the interval of values that a matched edge may take if it is to remain a matched edge. For any such edge $e_m(r_\alpha, t_\beta)$, the interval can be described as $[w_{m\alpha\beta} - \varepsilon_m, +\infty)$, where $w_{m\alpha\beta}$ is the edge weight of $e_m(r_\alpha, t_\beta)$ and ε_m is the *tolerance margin* that the weight can decrease without violating the optimality of the current matching solution. It is safe to increase the weight as this is a maximization problem. We say a matched edge is *hidden* if its weight has decreased so as to no longer form part of a matching solution.

Lemma 4.1. *With the resultant matching solution M_0 and the bipartite graph of the Hungarian algorithm, if a matched edge $e_m(r_\alpha, t_\beta)$ is hidden, the Hungarian algorithm can be completed with one iteration rooted at exposed node r_α. When a new perfect matching solution M' exists, the labeling reduction of the root r_α satisfies $l(r_\alpha) - l'(r_\alpha) = m_0 - m'$.*

Proof. The proof is based on the Hungarian algorithm solution method. If one matched edge $e_m(r_\alpha, t_\beta)$ is hidden, the bipartite graph remains feasible, but the equality graph G_e loses one matched edge. Hiding $e_m(r_\alpha, t_\beta)$ exposes r_α and t_β, requiring an iteration to complete the Hungarian algorithm. All

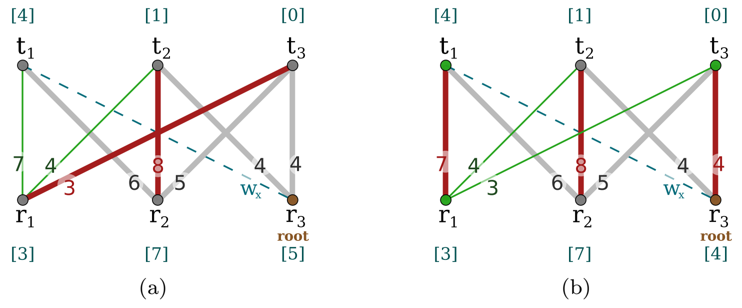


Figure 2: Interval analysis for a matched edge. (a) An objective matched edge hidden and assigned weight w_x; (b) New matching solution without the hidden edge.

other unexposed vertices are on the corresponding matched edges and, thus, are included in G_e, and moreover the sum of their labels is constant for all subsequent iterations. To grow G_e in bridging a new augmenting path[*], $l(r_\alpha)$ may decrease, but $l(t_\beta)$ remains constant because it is the end of the augmenting path and will be reached last. Therefore, $l(r_\alpha)$ is the only variable that can reduce the optimum. This proves that the reduction of $l(r_\alpha)$ is exactly the reduction of optimum from M_0 to M', namely, $l(r_\alpha) - l'(r_\alpha) = m_0 - m'$. □

Theorem 4.2 (**Matched Edge Interval).** *Hiding a matched edge from the Hungarian solution M_0 leads to a new solution $M^{'}$, and the labeling reduction $\varepsilon_m = m_0 - m^{'}$ at the root of the Hungarian tree is the tolerance margin for this element,* i.e., *the safe interval for a matched edge $e_m(r_\alpha, t_\beta)$ is $[w_{m\alpha\beta} - (m_0 - m^{'}), +\infty)$.*

Proof. Assigning weight $w = w_{m\alpha\beta} - \varepsilon_m$, where $\varepsilon_m = m_0 - m^{'}$, to edge $e_m(r_\alpha, t_\beta)$ results in optimum $m_0 - \varepsilon_m$ for the original matching M_0. Any new matching M' which does not contain $e_m(r_\alpha, t_\beta)$ also has an optimum of $m' = m_0 - \varepsilon_m$, and therefore both M_0 and M' are optimal matching solutions. Whenever the weight of edge $e_m(r_\alpha, t_\beta)$ satisfies $w \geq w_{m\alpha\beta} - \varepsilon_m$, we have $m_0 \geq m'$ implying that M_0 is optimal, otherwise M_0 would be substituted by M'. Thus, the maximum allowable interval for matched edge $e_m(r_\alpha, t_\beta)$ is $[w_{m\alpha\beta} - \varepsilon_m, +\infty)$, *i.e.*, $[w_{m\alpha\beta} - (m_0 - m^{'}), +\infty)$. □

Lemma 4.1 and Theorem 4.2 permit computation of the interval of a matched edge $e_m(r_\alpha, t_\beta)$ in the following way: First, hide $e_m(r_\alpha, t_\beta)$ from the bipartite graph and assign it an undecided weight w_x satisfying the constraint: $w_x < l(r_\alpha) + l(t_\beta)$. Next, let exposed vertex r_α be the root of a Hungarian tree and construct an augmenting path excluding $e_m(r_\alpha, t_\beta)$. The algorithm

[*]Although the definition of *augmenting path* is not necessary to understand Algorithm 3.1, we note that here we deviate from Lovász and Plummer [1986], pg. *xxxii*, where they use the term M-augmenting path.

Algorithm 4.1 Intervals of Matched Edges

```
Input:
    A matched edge $e_m(r_\alpha, t_\beta)$ and the corresponding resultant bipartite graph.
Output:
    Interval (lower bound $\xi_m$) for $e_m(r_\alpha, t_\beta)$.
1: Hide $e_m(r_\alpha, t_\beta)$ by assigning it an unknown weight $w_x$.
   Set $S = \{r_\alpha\}$, $T = \varnothing$.
2: If $N(S) = T$, update labels:
   $\delta = min_{x \in S, y \in Y-T, e(x,y) \neq e_m(r_\alpha, t_\beta)}\{l(x) + l(y) - w(x,y)\}$
   $l'(v) = \begin{cases} l(v) - \delta & if\ v \in S \\ l(v) + \delta & if\ v \in T \\ l(v) & otherwise \end{cases}$
   $l'(r_\alpha) + l'(t_\beta) - w_x > \delta \Rightarrow w_x < l'(r_\alpha) + l'(t_\beta) - \delta$.
   Update $\xi_m = l'(r_\alpha) + l'(t_\beta) - \delta$.
3: If $N(S) \neq T$, pick $y \in N(S) - T$.
   (a) If $y = t_\beta$, there must be an augmenting path $r_\alpha \to t_\beta$. Augment matching and terminate.
   (b) If $y$ matched, say to $z$, extend Hungarian tree: $S = S \bigcup \{z\}$, $T = T \bigcup \{y\}$.
   Go to step 2.
```

terminates when such a path is found that also generates a perfect matching. Because $l(t_\beta)$ stays unchanged but $l(r_\alpha)$ is decreased, w_x will decrease per iteration; the lower bound $w_{m\alpha\beta} - (m_0 - m^{'})$ occurs the moment a new perfect matching exists.

Figure 2 provides an illustratory example. Hiding matched edge $e(r_3, t_1)$ requires construction of a Hungarian tree rooted at newly exposed vertex r_3. Here $l(r_3)$ decreases while searching for an augmenting path, and a new matching solution $\langle e(r_1, t_1), e(r_2, t_2), e(r_3, t_3)\rangle$ replaces the original one in the augmenting path $r_3 \to t_3 \to r_1 \to t_1$. The reduction of labeling for r_3 is $5 - 4 = 1 = \varepsilon_m$, and the interval for $e(r_3, t_1)$ is $[8, +\infty)$.

4.2 Interval Analysis for Unmatched Edges

A similar analysis can be carried out for unmatched edges. An unmatched edge $e_u(r_\alpha, t_\beta)$ has an interval $(-\infty, \xi_u]$, where the upper bound ξ_u reflects the maximum utility value it may take while remaining an unmatched robot and task pair.

Lemma 4.3. *In the resultant bipartite graph of the Hungarian algorithm, the weight of any unmatched edge $e_u(r_x, t_y)$ can be increased to the sum of two associated labeling values $l(r_x) + l(t_y)$ without affecting the optimum assignment.*

Proof. An increment of this form does not violate the feasibility of the bipartite graph, nor does the modification remove any admissible edges from G_e. Therefore, the original matching in G_e remains perfect. □

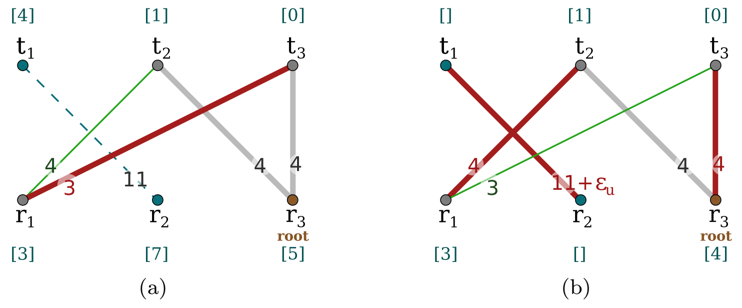


Figure 3: Interval Analysis for an unmatched edge. (a) Hide all associated edges of objective unmatched edge; (b) New matching solution formed with the objective edge and matching solution in auxiliary bipartite graph.

However, the upper bound in Lemma 4.3 is not tight. For example, in edge $e(r_2, t_1)$ of the graph in Figure 1(c), the weight can safely increase to 12 rather than $4 + 7 = 11$ (12 is the upper bound because greater weights result in optimal matching $\langle e(r_1, t_2), e(r_2, t_1), e(r_3, t_3)\rangle$), and there is a unit tolerance margin. Next, we show how to find the tolerance margin ε_u and further improve the upper bound. We redefine the interval as $(-\infty, l(r_\alpha) + l(t_\beta) + \varepsilon_u]$. It is important to note, this also shows that all unmatched edges, whether in G_e or not, can be treated uniformly once they become admissible.

To obtain ε_u for unmatched edge $e_u(r_\alpha, t_\beta)$, hide $e_u(r_\alpha, t_\beta)$ and all other edges incident to vertices r_α and t_β from the resultant bipartite graph. This yields a bipartite graph with $n - 1$ vertices in each partition. We term this new bipartite graph the *auxiliary* bipartite graph G_a. Notice that this auxiliary bipartite graph is associated with a particular edge, and that the auxiliary bipartite graph has only $n - 2$ matched edges. It therefore requires the addition of one edge for a matching solution.

Theorem 4.4 (**Unmatched Edge Interval**). *Any unmatched edge $e_u(r_\alpha, t_\beta)$ in the Hungarian resultant bipartite graph, has interval tolerance margin $\varepsilon_u = m_0 - (m_a + l(r_\alpha) + l(t_\beta))$, where m_0 is the optimum of the original solution, and m_a is the optimum of the auxiliary bipartite graph associated with $e_u(r_\alpha, t_\beta)$. The allowable interval for edge $e_u(r_\alpha, t_\beta)$ is $(-\infty, m_0 - m_a]$.*

Proof. For an arbitrary unmatched edge $e_u(r_\alpha, t_\beta)$ and its associated auxiliary bipartite graph G_a of size $n - 1$, we add $e_u(r_\alpha, t_\beta)$ to the matching solution M_a of G_a. This forms a new matching M' of size n. If the weight of edge $e_u(r_\alpha, t_\beta)$ satisfies $w_{u\alpha\beta} > m_0 - m_a$, then the matching M' containing $e_u(r_\alpha, t_\beta)$ must satisfy $m' = w_{u\alpha\beta} + m_a > m_0$. But this contradicts the fact that the original matching M_0 was perfect. This proves that the upper bound for unmatched edge $e_u(r_\alpha, t_\beta)$ is $m_0 - m_a$, and that the allowable interval is $(-\infty, m_0 - m_a]$. □

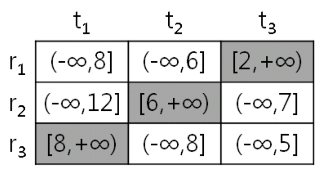
Consider Figure 3 as an example. Hiding $e(r_2, t_1)$ and all edges incident to r_2 and t_1 leaves auxiliary bipartite graph G_a containing vertices r_1, r_3, t_2,

Algorithm 4.2 Intervals of Unmatched Edges

```
Input:
    An unmatched edge $e_u(r_\alpha, t_\beta)$ and the corresponding resultant bipartite graph.
Output:
    Interval (upper bound $\xi_u$) for $e_u(r_\alpha, t_\beta)$.
1: Assume $e(r_\alpha, mate(r_\alpha))$, $e(mate(t_\beta), t_\beta)$ are matched edges, then set $S = \{mate(t_\beta)\}$, $T = \varnothing$.
2: Hide $e_u(r_\alpha, t_\beta)$ and all other edges incident to vertices $r_\alpha$ and $t_\beta$, and obtain the auxiliary bipartite graph $G_a$.
3: In $G_a$, if $N(S) = T$, update labels:
   $\delta = min_{x \in S, y \in Y-T}\{l(x) + l(y) - w(x,y)\}$
   $$l'(v) = \begin{cases} l(v) - \delta & if\ v \in S \\ l(v) + \delta & if\ v \in T \\ l(v) & otherwise \end{cases}$$
4: In $G_a$, if $N(S) \neq T$, pick $y \in N(S) - T$.
   (a) If $y = mate(r_\alpha)$, there must be an augmenting path $mate(t_\beta) \rightarrow mate(r_\alpha)$. Augment matching and go to step 5.
   (b) If $y$ matched, say to $z$, extend Hungarian tree: $S = S \bigcup \{z\}$, $T = T \bigcup \{y\}$. Go to step 3.
5: $\xi_u = m_0 - m_a$.
```

$*$ Definitions:

- $mate(v)$ is the other ending vertex with respect to vertex v;
- m_0 is optimum of the original solution, m_a is optimum of G_a.



	t_1	t_2	t_3
r_1	$(-\infty,8]$	$(-\infty,6]$	$[2,+\infty)$
r_2	$(-\infty,12]$	$[6,+\infty)$	$(-\infty,7]$
r_3	$[8,+\infty)$	$(-\infty,8]$	$(-\infty,5]$

Figure 4: An example of an interval matrix

t_3 and associated edges (see Figure 3(a)). The Hungarian tree is created with root at newly exposed vertex r_3, and finally the matching solution M_a of G_a is $\langle e(r_1,t_2), e(r_3,t_3)\rangle$, as shown in Figure 3(b). The tolerance margin and allowable interval for $e(r_2,t_1)$ are $\varepsilon_u = m_0 - (m_a + l(r_2) + l(t_1)) = 1$ and $(-\infty, 12]$, respectively.

4.3 Interval Hungarian Algorithm

Combining the interval analysis of matched and unmatched edges, we have the Interval Hungarian algorithm described in Algorithm 4.3. Figure 4 shows the corresponding intervals for the assignment matrix in Figure 1(a).

Algorithm 4.3 Interval Hungarian Algorithm

```
Input:
   A resultant bipartite graph from running Algorithm 3.1.
Output:
   A matrix $mx_{itv}(n,n)$ of upper and lower bounds on each interval.
1: $mx_{itv}(n,n)$ = NULL.
2: foreach edge $e(i,j)$ in bipartite graph do
     if $e(i,j)$ is matched then
       compute interval $I(i,j)$ with Algorithm 4.1;
       $mx_{itv}(i,j) = I(i,j)$;
     else
       compute interval $I(i,j)$ with Algorithm 4.2;
       $mx_{itv}(i,j) = I(i,j)$;
     end if
   end foreach
3: return $mx_{itv}$.
```

4.4 Complexity Analysis

Since the Hungarian algorithm has a computational complexity of $O(n^3)$, our Interval Hungarian algorithm has a worst-case time complexity of $O(n^4)$. Obtaining the intervals for n matched edges, needs an extra $O(n \times n^2) = O(n^3)$

operations since for each edge requires the construction and search of a Hungarian tree, which costs $O(n^2)$. The computation of all the unmatched edges has a worst case cost of $O(n^4)$ since, for each unmatched edge, we do the same searching iteration in G_a (it has a size of $n-1$ and thus needs a computational complexity of $O((n-1)^2)$). There are $n^2 - n$ unmatched edges, producing the resultant $O((n^2 - n) \times (n-1)^2) = O(n^4)$.

The time complexity can be further reduced if a parallel mechanism is available, as is the case when the multiple robots form a distributed computing system. This is because the computation of each interval does not depend on the computation of any others, *i.e.*, each interval only depends on the resultant bipartite graph output from the Hungarian algorithm. If k processors are provided, each processor should be assigned with n^2/k intervals given the complete task involves n^2 intervals. Because each interval is independent from all others and requires $O(n^2)$ time, the total running time is $O(n^4/k)$. When $k \geq n$, the overall time complexity is still at most $O(n^3)$. This is very useful in the multi-robot task assignment problem since the multi-robot system provides exactly n processors.

5 Quantifying the Effect of Uncertainty

When the Hungarian algorithm is applied to a matrix of expected utilities calculated from uncertain data (*e.g.*, using the mean of a utility distribution) one has little idea of the impact this uncertainty has on the resultant assignment. The output from the Interval Hungarian algorithm can be used to analyze the changes in the optimal allocation as changes are made to particular utility values. Next, we demonstrate that the algorithm can be used to estimate the likelihood that the calculated assignment will be sub-optimal.

5.1 Uncertainty Measurement for a Single Utility

Theorem 5.1 (**Uncertainty of a Single Interval**). *With regard to any specific single utility value, assuming other utilities are certain, the perfect matching solutions are identical if and only if any specific utility is within its allowable interval.*

Proof. Assuming other utilities in the assignment matrix are certain, taken together Theorems 4.2 and 4.4 prove that if the matching solution remains the same, then any specific utility must be within its allowable interval. To prove the converse: suppose the matching solutions are not identical, then there must be a new perfect matching M' that replaces the original one M_0 *i.e.*, $m' > m_0$. However, this cannot happen since any value within the interval must produce a matching solution with sum of weights which is less than or equal to m_0. $\square$

To analyze the effect of uncertainty on a specific utility in the assignment matrix, we assume the other values are certain. Given a probability density function $f(x)$ for this specific expected utility and an associated interval I as output from the algorithm, a *reliability score* that reflects the probability of yielding the current optimal solution is:

$$P_I = \begin{cases} \int_{\xi_m}^{+\infty} f(x), & \text{when } I = [\xi_m, +\infty) \\ \int_{-\infty}^{\xi_u} f(x), & \text{when } I = (-\infty, \xi_u]. \end{cases} \tag{1}$$

For applications in which robots are actively estimating quantities involved in producing $f(x)$, one may set some threshold $T \in [0, 1]$, so the robots only commit to an assignment if $P_I \geq T$, and instead opt to invest resources in reducing the uncertainty in the estimate when it is likely to have a major bearing on the particular assignment. High values of T will ensure the robots only commit to allocations that are robust to errors in estimates of the expected utility.

5.2 Uncertainty Measurement for Interrelated Utilities

The previous subsection gives an approach for quantifying the effect of uncertainty on the robot-to-task assignment when only one utility was uncertain. Most often, however, multiple utilities are uncertain and they may all be interrelated if they involve inference over the same underlying state variables. For example, a row in the assignment matrix represents all relevant expected utilities for a specific robot. A change that effects the performance of the robot (*e.g.,* low battery) effects all the entries in the row. Here we use the term *interrelated edges* to represent all directly related utilities in a single row or column. For the same assignment to be preserved, in spite of the n interrelated edges, there must be one and only one edge that remains matched and all the others should be unmatched.

Theorem 5.2 (**Uncertainty of Interrelated Intervals**). *Given a set of n interrelated edges, assume e_m is the matched edge with interval $[w_m - \varepsilon_m, +\infty)$, and e_{ui} are unmatched edges with intervals $(-\infty, w_{ui} + \varepsilon_{ui}]$, $i = 1, 2, \cdots, n-1$, then for any $\varepsilon' \leq \varepsilon_m$, the weight of e_m can be safely substituted with $w_m - \varepsilon'$, and the interval for e_{ui} becomes $(-\infty, w_{ui} + \varepsilon_{ui} - \varepsilon']$, $i = 1, 2, \cdots, n-1$.*

Proof. $\varepsilon' \leq \varepsilon_m$ indicates that new weight $w'_m = w_m - \varepsilon'$ is within the interval associated with edge e_m, thus a substitution of w'_m will not violate the matching solution. To prove the interval form for e_{ui}, $i = 1, 2, \cdots, n-1$: suppose that m'_0 is the optimum with substituted weight w'_m, then we have $m'_0 = m_0 - \varepsilon'$. From Theorem 4.4, the interval for e_{ui} is $(-\infty, m'_0 - m_a]$, which can be substituted with $(-\infty, m_0 - \varepsilon' - m_a]$. Because $m_0 - m_a = w_{ui} + \varepsilon_{ui}$, the interval for e_{ui} becomes $(-\infty, w_{ui} + \varepsilon_{ui} - \varepsilon']$. □

Notice, Theorem 5.2 exploits the mutual exclusion property of interrelated unmatched edges: at any time one and only one interrelated unmatched edge can possibly become matched. This means that as the matched edge's weight is decreased, one unmatched edge moves closer to its interval's upper-bound. However, as the matched edge's value decreases, the bounds of all the other

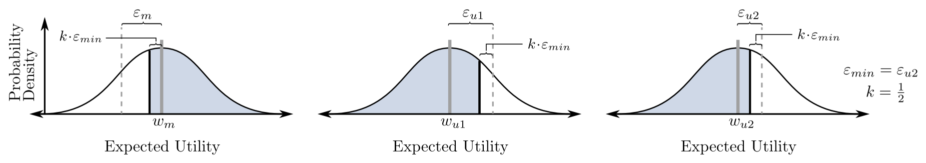


Figure 5: Reliability scores are determined for interrelated edges by integrating the shaded areas. This example show three edges, each with utility uncertainty represented by a Gaussian distribution. The leftmost edge is matched while the other two are unmatched.

unmatched edges decrease too. To allow the simultaneous occurrence of interrelated matched edge and unmatched edges, we compromise between them: intervals of interrelated unmatched edges shrink by ε', while the interval for the interrelated matched edge shrinks by $\varepsilon_m - \varepsilon'$. The following 4 step method determines the uncertainties of interrelated edges:

1. Determine ε_{min} from all interrelated edges:
   $\varepsilon_{min} = min(\varepsilon_m, \varepsilon_{ui}), \quad i = 1, 2, \cdots, n-1.$

2. Determine each interrelated interval I_i:

$$I_i = \begin{cases} [w_m - k \cdot \varepsilon_{min}, +\infty), & i = 0; \\ \\ (-\infty, w_{ui} + \varepsilon_{ui} - k \cdot \varepsilon_{min}], & i = 1, 2, \cdots, n-1. \end{cases}$$

   Note: I_0 represents the interval for the matched edge. $k \in [0, 1]$ is a coefficient which affects a degree of scaling between matched and unmatched intervals.

3. Determine reliability scores:

$$P_{Ii} = \begin{cases} \int_{\xi'_m}^{+\infty} f(x), \ (\xi'_m = w_m - k \cdot \varepsilon_{min}), & i = 0; \\ \\ \int_{-\infty}^{\xi'_{ui}} f(x), \ (\xi'_{ui} = w_{ui} + \varepsilon_{ui} - k \cdot \varepsilon_{min}), & \text{otherwise.} \end{cases} \tag{2}$$

4. Determine reliability level:
   The assignment is reliable when $P_{Ii} \geq T$, and unreliable otherwise.

Thus, the margin of tolerance shrinks for all interrelated edges. This approach uses a parameter k to balance this shrinkage between interrelated matched and unmatched edges. Equation 2 implies that when $k \neq 0$, both matched and unmatched edges shrink by values associated with k: every unmatched edge shrinks by $\varepsilon' = k \cdot \varepsilon_{min}$, and the unique matched edge shrinks by a complementary value of $\varepsilon_m - \varepsilon^{'}$. In Step 4, the uniform reliability threshold T requires that every edge be sufficiently reliable ($P_{Ii} \geq T$, $\forall i$). This means that a whole set of interrelated edges can be judged reliable enough by examining the lowest score, *i.e.,* $\min\{P_{Ii}\}$. Therefore, we want to maximize the lowest score, and k may be obtained by:

$$k = \arg\max_{k'} \Big(\min\{k^{'}\varepsilon_{min},\ \varepsilon_{u1} - k^{'}\varepsilon_{min},\ \varepsilon_{u2} - k^{'}\varepsilon_{min},\ \cdots\}\Big), \qquad (3)$$

where $k\varepsilon_{min}$ is the new tolerance margin for the interrelated matched edge, and the others are for the set of interrelated unmatched edges. Obviously, when $k \to 0$, the new tolerance margin of the interrelated matched edge approaches zero—so that the uncertainty of the matched edge is deemed unimportant—while the interrelated unmatched edges have almost no reduction from their original margins. In this case it is very likely that the lowest score will be from the matched edge (determined via the integral of only the right half of the probablity density function). Thus, in such cases, the unmatched edges will play little role in determining whether the assignment is robust. The other extreme forms a contrast: as $k \to 1$, there must be an interrelated unmatched edge that has new tolerance margin of zero, which likely results in the lowest score with similar drawback.[†]

The use of a single parameter is expected to be most effective for Gaussian-like distributions where the mean and medians of the distributions are close. Figure 5 illustrates the reliability scores (shaded area) under a Gaussian distribution density for the matched and unmatched edges, respectively. This method can measure the uncertainties for a horizontal row or a vertical column in the assignment matrix, assuming other non-directly interrelated rows or columns are known with certainty. The uncertainty involving multiple rows or columns is complex and beyond the scope of this paper. Although the approach does not address the full problem in which the sensitivity an arbitrary sub-matrix depends on the knowledge of all the other intervals, by permitting variation interrelated values moves beyond a mere "perturbation" like analysis. Perhaps most important, experiments in Sections 6 and 7 illustrate that the Interval Hungarian algorithm is effective for realistic tasks.

5.3 Measuring the Uncertainty in Multi-robot Systems

In this subsection, we design an approach using Interval Hungarian algorithm to measure the uncertainty problem in multi-robot systems. As discussed in Subsection 4.4, the computation of intervals may be separated and distributed due to their independence property. It is natural that a multi-robot system should take advantage of this feature to parallelize computation of the intervals. Moreover, the robots can actually delay this computation so that it is executed

[†]This conclusion has been verified in our experiments involving 3 uncertainty-robots, by testing $k \in \{0, 0.1, \cdots, 1\}$. The results show that the max-min optimum typically results when $0.4 \le k \le 0.6$. Obtaining an analytical solution is unlikely, we therefore adopt the empirical value of $k = 0.5$ in our experiments. This is reasonable because a margin of at least $\frac{1}{2}\varepsilon_{min}$ for each edge is guaranteed. The imperfection of k can also be compensated for by a slightly higher value of T in practice.

only when there are relevant queries for assessment of the effect of uncertainty on the optimal allocation.

Algorithm 5.1 describes the implementation of the uncertainty measurement using Interval Hungarian algorithm for task allocation in multi-robot systems. In Step 3 of Algorithm 5.1, the decision-making robot distributes the Hungarian optimal solution and corresponding resultant bipartite graph to all peer robots, and then each robot is responsible for computing its interrelated intervals when it encounters uncertainty (we denote such a robot an *uncertainty-robot*). The algorithm permits multiple uncertainty-robots and need not assume that all participating robots are uncertain. The preceding derivation of a method for computing interrelated intervals on one robot is based on the assumption that all other utilities are certain (or remain fixed); the existence of multiple simultaneous uncertainty-robots introduces inaccuracy in measuring the effect of uncertainty with regard to some utilities. To handle this, a conservative approach is adopted: we raise the reliability threshold T when more than one concurrent uncertainty query occurs, *e.g.*,

$$T = T_0 + \exp[-(n - n_x)] \cdot (1 - T_0), \quad (1 < n_x \leq n) \tag{4}$$

where n, n_x and T_0 are the total number robots, the number of uncertainty-robots, and the initial threshold for a single uncertainty-robot, respectively. The overall uncertainty increases super-linearly with increasing n_x. Although many other similar forms are also possible, the exponential function was chosen based on the following intuition: since the uncertainty for each individual robot is approximately independent, one wishes to “multiply” these uncertainties.

As before, we require the reliability scores of all interrelated intervals of every concurrent uncertainty-robot to be greater than the threshold T, as described in Step 5 of Algorithm 5.1.

6 A Demonstration in Simulation

This section introduces a simple multi-robot task allocation problem which illustrates that the Interval Hungarian algorithm allows the effect of uncertainty on the optimal allocation to be quantified. We consider the problem of dispatching a group of homogeneous robots to a set of destination locations. We select this task because it and its variations have been used in the literature for the purpose of evaluating task assignment methods (*e.g.,* Berhault *et al.* [2003] and subsequent papers by Koenig’s group). Additionally, the source of uncertainty and the means for actively estimating this uncertainty has been well-studied.

Other tasks (*e.g.,* cooperative box pushing experiments [Gerkey and Matarić, 2002], Parker’s Cooperative Multi-robot Observation of Multiple Moving Targets (CMOMMT) task) [Parker, 1998] have different forms of uncertainty, but once a utility matrix is constructed for purposes of robot task-allocation, the method applies directly. The source of the uncertainty will depend on the particular problem domain. For example, if the robots are observing moving targets, two primary aspects contribute to the utility calculation: a quality measure and

Algorithm 5.1 Optimal assignment and uncertainty measurement in multi-robot systems

```
Data:
    An $n \times n$ utility matrix.
Result:
    Assignment decision for the multi-robot system, as well as assignment reliability levels for the uncertainty-robots.

 {/*do following procedure at every time period $t$*/}
1: Decision-making robot $r_d$ runs Hungarian algorithm, gets solution $M_0$ and resultant bipartite graph;
2: $r_d$ broadcasts $M_0$ and resultant bipartite graph to all other robots;
   {/*do "foreach" below in distributed fashion, $i, j \in [1, n]$*/}
3: foreach robot $r_i$ do
     if uncertainty query then
       compute the interval $I_{ij}$ for each interrelated utility;
       notify $r_d$ to count total number of uncertainty-robots;
     end if
   end foreach
4: $r_d$ decides reliability threshold $T$ based on the number of uncertainty-robots, and broadcasts $T$;
5: foreach uncertainty robot $r_i^{'}$ do
     compute reliability $P_{Iij}$ for each interrelated utility;
     if any $P_{Iij} < T$ then
       improve the certainty before committing assignment;
     end if
   end foreach
```

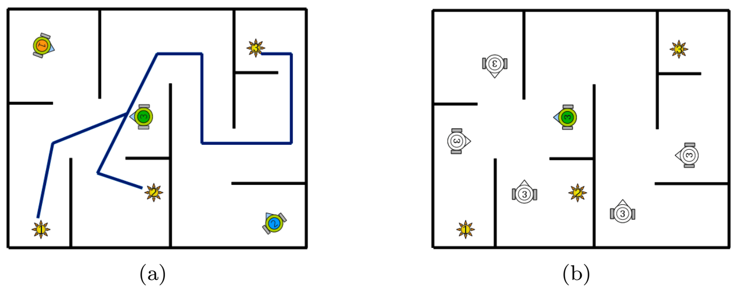
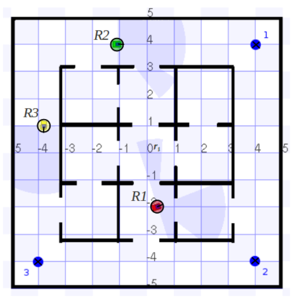


Figure 6: Multi-robot task assignment problem in which location uncertainty results in a distribution of utility estimates. (a) Planned paths to each task from the central pose estimate of Robot #3 are shown in blue; (b) Robot #3 is uncertain of its location within the environment. In this instance, the robot has six pose estimates including the correct central one.

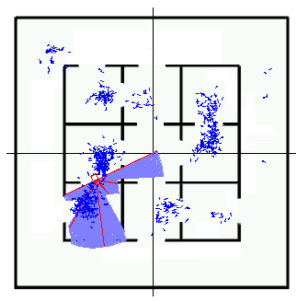
a cost metric. Generally both will involve a degree of uncertainty, but this is especially true when no model is provided for future movement of the targets. Quality itself would need to be calculated as an expectation over potential target movements, whereas costs depend on sensor particularities, navigational ability, *etc.*

In the multi-robot dispatching problem we use further in this paper, the utility matrix is a negative estimation of the distance from current location to goal location, and the uncertainty in the assignment problem results from localization error. Each robot attempts to localize itself in the environment by employing an particle filter-based approach and a given map [Fox, 2001]. Particles are clustered and the weighted means of these clusters taken as pose hypotheses. A planner is used to estimate the cost from estimated location to a given goal location. Figure 6 shows three robots being dispatched to three task locations in a maze-like room. The robot labeled #3 is located in the center of the maze; from that location it employs a wavefront planner (i.e., a breadth-first search over a discretization of the environment) to estimate costs to the three tasks. The line segments in Figure 6(a) are the planned paths from the (true) central location and the path cost is the total length of each path. When the robot is uncertain about its location (*e.g.*, as shown in Figure 6(b), where robot #3 has a set of possible poses) then the result is a distribution of possible utility values.

The simulation environment is illustrated in Figure 7(a) within the Stage simulator [Gerkey *et al.*, 2003]. The maze was designed to involve some symmetry so as to provide multiple uncertain poses during localization. Robots were initialized at random positions within the maze and dispatched to the task locations shown as blue dots in Figure 7(a). A subset of the robots, called *localized-robots*, obtain ground-truth poses from the simulator, while the remainder are uncertainty-robots which use a simulated scanning laser ranging sensor and a map (identical to the simulation environment) along with the localization



(a)



(b)

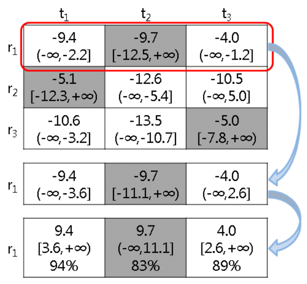
Figure 7: (a) Multi-robot task assignment and commitment in a maze; (b) Uncertain hypothesis for uncertainty-robot (robot $R1$ in red).

algorithm and thus have uncertain poses. Figure 7(b) shows the particles for an uncertainty-robot. A *path cost distribution* shows the effect of the uncertainty on a robot's utility estimates; it is obtained by computing path lengths from all available hypothesis of uncertainty-robot to a specific task location.

For the purposes of demonstration, we consider a scenario using 3 robots as shown in Figure 7. Of the 3 robots, only one (Robot $R1$, red color) is an uncertainty-robot and the others are localized-robots. Figure 8 is the corresponding captured localization uncertainties at time $t = 105s$. Figure 8(a) is the assignment matrix with the optimal solution (shaded cells) and safe intervals for all the utilities. Figures 8(b) — 8(d) are the path cost distributions associated with corresponding hypotheses. In each sub-figure, the bar with highest probability (highlighted with stripes) is the path cost for the current most-likely localization result; therefore it is used in the assignment matrix. Other bars are the path costs for all other uncertain hypothesis. Remember, only Robot $R1$ has imperfect localization and the intervals for the interrelated utilities, the first row Figure 8(a), are $\langle(-\infty, -2.2], [-12.5, +\infty), (-\infty, -1.2]\rangle$; thus we can compute its tolerance margin $\varepsilon_m = -9.7 - (-12.5) = 2.8$. Following Algorithm 5.1, we get the allowable interrelated intervals $\langle(-\infty, -3.6], [-11.1, +\infty), (-\infty, -2.6]\rangle$ for Robot $R1$. The intervals are flipped from negative to positive to fit the real path cost distributions: $\langle[3.6, +\infty), (-\infty, 11.1], [2.6, +\infty)\rangle$.

In Figures 8(b) — 8(d), the bars within interrelated intervals are solid red (darker) and the others are in pink (lighter). The calculated results of P_I are $\langle 94\%, 83\%, 89\%\rangle$ and we use a reliability threshold $T = 80\%$, therefore all utilities are considered reliable and we can trust the assignment solution.

Figure 9 is another scenario which illustrates an "unreliable assignment solution". The reliability levels of interrelated utilities are $\langle 82\%, 39\%, 55\%\rangle$ as shown in Figure 9(c). There are two utilities below the threshold, thus, we consider the assignment solution unreliable. In fact, from Figure 9(a) and 9(b) we see that if the localization of Robot $R1$ is at its true location then the assignment solution should be $r_1 \rightarrow t_3$ (Robot $R1$ is assigned to the task location #3), $r_2 \rightarrow t_1$ and $r_3 \rightarrow t_2$. This case illustrates the sensitivity of the

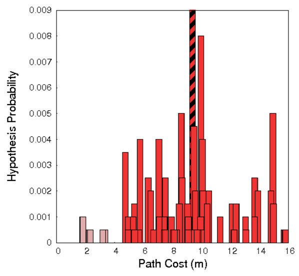


	t_1	t_2	t_3
r_1	-9.4 (-∞,-2.2]	-9.7 [-12.5,+∞)	-4.0 (-∞,-1.2]
r_2	-5.1 [-12.3,+∞)	-12.6 (-∞,-5.4]	-10.5 (-∞,5.0]
r_3	-10.6 (-∞,-3.2]	-13.5 (-∞,-10.7]	-5.0 [-7.8,+∞)

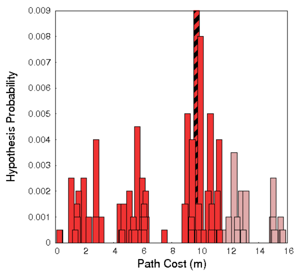
r_1	-9.4 (-∞,-3.6]	-9.7 [-11.1,+∞)	-4.0 (-∞,2.6]

r_1	9.4 [3.6,+∞) 94%	9.7 (-∞,11.1] 83%	4.0 [2.6,+∞) 89%

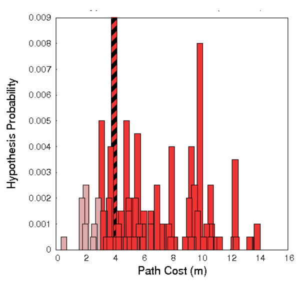
(a)



(b)

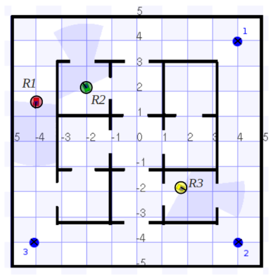


(c)

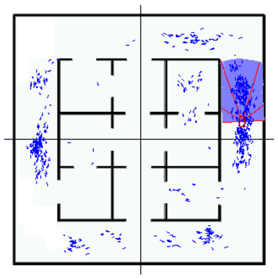


(d)

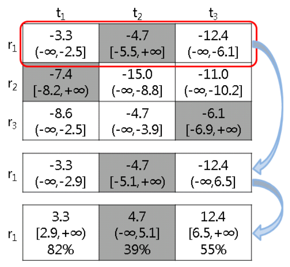
Figure 8: (a) Uncertainty analysis of interrelated intervals for robot $R1$; (b) — (d) Hypothesis distributions of interrelated utilities captured at the 105$^{\text{th}}$ second.



(a)



(b)

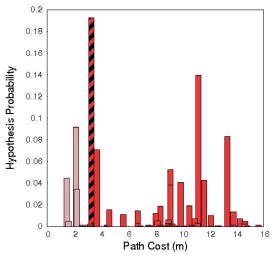


	t_1	t_2	t_3
r_1	-3.3 (-∞,-2.5]	-4.7 [-5.5,+∞]	-12.4 (-∞,-6.1]
r_2	-7.4 [-8.2,+∞)	-15.0 (-∞,-8.8]	-11.0 (-∞,-10.2]
r_3	-8.6 (-∞,-2.5]	-4.7 (-∞,-3.9]	-6.1 [-6.9,+∞)

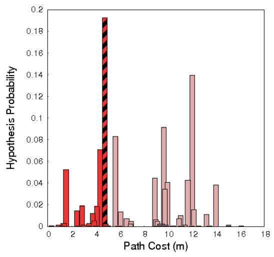
r_1	-3.3 (-∞,-2.9]	-4.7 [-5.1,+∞)	-12.4 (-∞,6.5]

r_1	3.3 [2.9,+∞) 82%	4.7 (-∞,5.1] 39%	12.4 [6.5,+∞) 55%

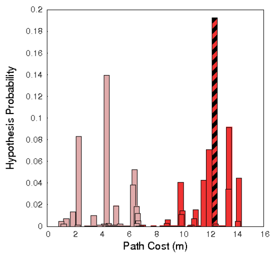
(c)



(d)



(e)



(f)

Figure 9: (a) — (b) Multi-robot assignment simulation captured at 76th second; (c) — (f) Uncertainty analysis of interrelated utilities.

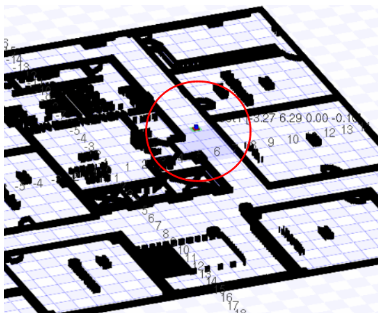
assignment solution to uncertainty and the algorithm's output suggests refining of the localization before committing to an unreliable assignment.

7 Experiments

By measuring and triggering reduction of the uncertainty only when it is applicable to the task-allocation instance at hand, the Interval Hungarian algorithm has an important advantage over the standard approaches. In this section we describe experiments we conducted using the Interval Hungarian algorithm to quantify the uncertainty which results from localization error. The experiments show that the risks of mis-allocation can be reduced. Results both in simulation and with physical robots are presented and we include a comparative analysis.

7.1 Simulation Setup

Simulation experiments were conducted in the Stage simulator [Gerkey *et al.*, 2003]. Different from the simple demonstration described in Section 6, the simulation here is also carried out in a distributed fashion: each simulated robot localizes with an identical map on a local machine, and performs the steps described in Algorithm 5.1. Figure 10(a) shows the simulated robot localizing


(a)

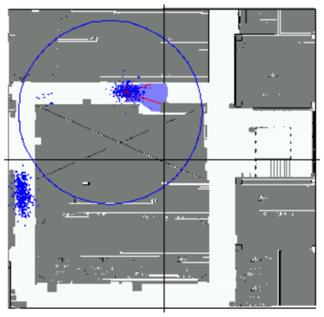

(b)

Figure 10: Experiments in Stage simulator. (a) Simulation of an uncertainty-robot localizing in our research building; (b) Localization particles generated from simulation.

in a floor legend of our research building. The corresponding localization results and particles are shown in Figure 10(b).

The communications among task committing robots are implemented via UDP. Each robot runs a UDP server and listens to the messages sent by teammates. A single robot was arbitrarily selected to be the decision-making robot and thus perform the global assignment. The decision-making robot employs two types of messages: (1) command messages, (2) query messages. The command messages are used to communicate actions to individual robots (actions include stop, go, or hover around the current location for better localization). The query message is for obtaining a result, such as the reliability scores or accumulated running time, *etc.*, which are computed locally by the member robots distributed across the machines.

We obtain localization hypothesis information, as well as the planned paths, by extracting data directly from a widely used particle filter localization method (see Fox [2001]). The data are stored locally and are processed on arrival of a UDP query message. The localization system plans global paths using a *wavefront* planner and plans the local paths via the *VFH+* (Vector Field Histogram) planner [Ulrich and Borenstein, 1998]; implementations of both were obtained via *player* [Vaughan and Gerkey, 2006]. The planners return a series of way-points which connect the current pose to the goal pose and a robot follows these way points to reach the assigned task location. The path cost from a pose hypothesis is computed by summing up all the path-segments among its corresponding way-points.

The uncertainty level of a robot reflected its localization speed (*i.e.*, the quantity and convergence of particles), which we can manipulate for experimental purposes mainly by tuning the variance of returned laser range data in the simulator. The greater variances represent greater uncertainty of sensed ranges, inducing additional uncertainty in the localization. Thus, a delayed localization process with slow particle convergence can be achieved by using a large laser range variance, whereas a fast localization with quick particle convergence can be achieved by using a small variance. In addition, the initial pose estimate and variance of initial pose estimate for the robot may also be used to control the localization accuracy and speed. By adjusting these parameters, we can easily obtain mixtures of uncertainty-robots and localized-robots.

7.2 Physical Robot Experiment Setup

Since the simulation operated in distributed manner, the drivers produced were equally applicable to physical robots, making it easy to transfer from simulation to physical robot experiments.

The robots we used are the *create* type made by iRobot, as shown in Figure 11(a). A Hokuyo URG-04LX-UG01 laser range sensor is mounted on each robot which records range data up to $\sim$5m and an ASUS EEE netbook is carried by each robot to compute all data including the communication, utility estimation, interval analysis, logging, *etc.*

The map used in the physical robot experiments was drawn to scale with the dimensions of our building. The robots started with extreme initial uncertainty. Since the highly symmetric environment resulted in considerable time being spent to localize, we added distinguishing features along some corridor walls. With a few obstacles and with enough travel time, our robots finally localize themselves regardless of the level of initial uncertainty. This addition to the environment facilitates fair analysis of the allocation results: a complete failure of task commitments due localization failure would artificially inflate the benefit of the proposed algorithm.

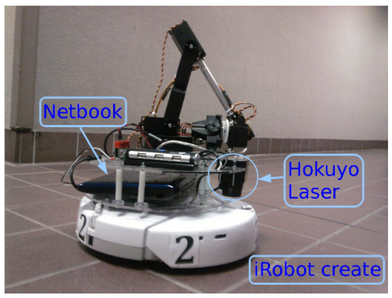
One difference between simulation and physical experiments is that, the localization technique cannot obtain the ground-truth positions of physical robots as in simulation. We addressed this issue by putting milestone-marks along the corridors and estimating the poses according to the scale between the map and real environment.

The results from the physical robot experiments are close to those reported from our simulation trials. This similarity in observed performance is due, at least in part, to the extensive work we did in tuning and testing the localization parameters as well as improvements in the laser drivers and client programs.

Localization results, particle convergence, planned paths, as well as the assignment consequences, are observed through the *playernav* utility tool, which is also available as part of the *player/stage* suite. Figures 11(c)—11(e) show the observed localization information and task allocations using *playernav*.

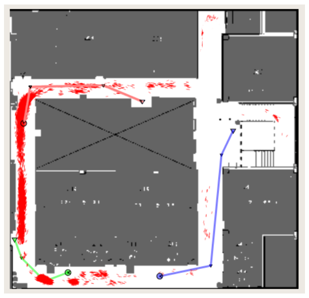
7.3 Experimental Procedure

Both simulation and physical robot experiments focused on measuring and comparing the results for the 3-robot way-point task assignment problem under location uncertainty. We conducted repeated experiments by placing all task committing robots in random places within the corridors, dispatching them

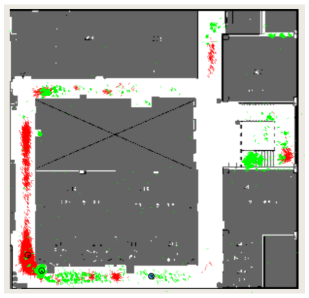


(a)

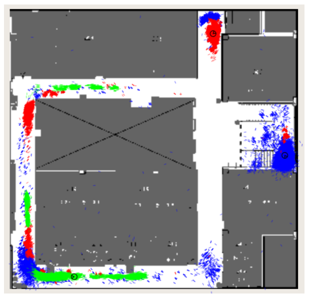
(b)



(c)



(d)



(e)

Figure 11: The setup for our physical robot experiments. (a) An *iRobot* create robot with Hokuyo laser and EEE netbook; (b) An uncertainty-robot localizing in our research building; (c)–(e) *playernav* showing particles with 1, 2, 3 uncertainty robot(s), respectively ((c) shows planned paths to allocated tasks).

to the nearest target locations, and following Algorithm 5.1 for uncertainty measurement and task allocation.

We have tested the assignment performance using Interval Hungarian algorithm with 1, 2 and 3 uncertainty-robot(s), respectively, and compared the results with the control experiments employing the standard Hungarian algorithm instead. More specifically, for each round of experiment (either simulation or physical robot experiment), the following procedure was carried out.

First, for each localized-robot (if any), we provide the correct initial pose, small initial pose variance and small laser range variance. Moreover, we start it a few seconds early permitting it to completely localize itself. This is further verified by a human operator.

If the Interval Hungarian algorithm is being used, for every 5 seconds each uncertainty-robot computes the current utility (negative path cost of current localization) and checks the reliability scores P_{Ii}s and compares them with the given threshold T. If any score is below the threshold, the uncertainty-robot refuses the assignment and broadcasts this refusal to the whole team so as to ensure all other (reliably) localized robots wait until all scores are satisfactory. The robot then performs a short-distanced (randomly directed) movement back and forth to better localize itself until all P_{Ii}s overcome the given threshold. The robot then resumes the task allocation commitment. Otherwise, if the Hungarian algorithm is being used instead, every 5 seconds the robots simply compute the immediate utilities and make an optimal assignment. The times of task assignment flips due to changing localization poses and the time for completion of the task allocations are recorded.

Finally, when the assigned tasks are performed, they are evaluated by comparison to a robot with the original ground-truth assignment result, and we regard this a success if they are identical. Otherwise, it is regarded a failure either due to mis-allocation from unreliable estimates, or task non-accomplishment due to incorrect final localizations.

7.4 Results and Analysis

The task accomplishment qualities of experiments described above are shown in Table 1. We count the success and failure times and present the data as percentages.

Note that there are two categories of failures: one is the discrepancy of final allocations (mis-allocations) comparing with the initial ground-truth assignment, and the other is failure due to the eventual wrong localization of robots, which usually results in incomplete arrival of the robot at the destination.

These results show that the Interval Hungarian algorithm (in both simulation and physical experiments) can greatly reduce the risk of task mis-allocations when utility estimations are unreliable. This is particularly true with increasing numbers of uncertainty-robots. The algorithm even performs better than immediate allocation directly using the Hungarian algorithm, although the total percentage of successes is reduced when compared to experiments involving fewer uncertainty-robots. This can be observed from the column of “Successes”

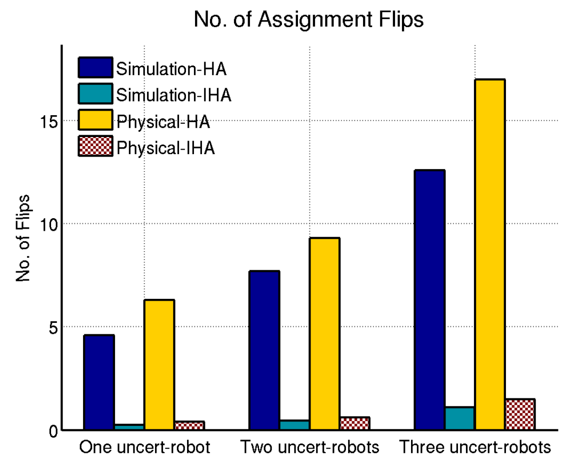
Table 1: Task Accomplishment Quality Comparison

Uncertainty#	Type	Method	Successes	Failures (Mis-allocation)	Failures (Localization)
One	Simulation	HA	55%	40%	5%
		IHA	85%	5%	10%
	Physical	HA	50%	35%	15%
		IHA	70%	10%	20%
Two	Simulation	HA	30%	55%	15%
		IHA	75%	15%	10%
	Physical	HA	35%	45%	20%
		IHA	75%	10%	15%
Three	Simulation	HA	15%	70%	15%
		IHA	65%	15%	20%
	Physical	HA	20%	60%	20%
		IHA	60 %	20%	20%

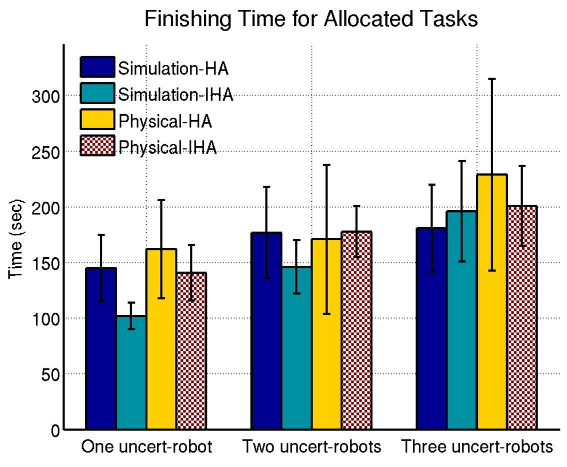
Each row of data is generated from 20 sets of experiments. $T = 70\%$, number of particles is 10^4, laser range variance is $2m$ for uncertainty-robots, and $0.2m$ for localized-robots.

and the percentages of *IHA* are always higher than those of *HA* including data for both simulation and physical experiments. To consider relative performance, one may take the ratio of percentages of successes *IHA*:*HA*. Both methods are less successful when greater numbers of uncertainty-robots are involved, but the ratio generally increases. Note that the exception is for *IHA* in physical experiments, where the percentage of sucesses first goes up and then comes down. We believe this is due to both system errors and localization failures; the failures due to localization occur more-or-less randomly, but with higher frequency in physical experiments than in simulation.

We also recorded the times of task allocation flips and the total time to finish all assigned tasks in each experiment. These data are illustrated in Figure 12. Figure 12(a) shows that Interval Hungarian algorithm greatly reduces the oscillatory allocation flips. This is because Interval Hungarian algorithm always evaluates each uncertain utility and refuses the unreliable assignments. The Hungarian algorithm simply assigns tasks based on the instantaneous localization results, and flips frequently as the localization estimates of individual uncertainty-robots change. Figure 12(b) shows the finishing time to accomplish all the tasks. Generally, the time costs of experiments using Interval Hungarian algorithm is slightly less than those of using Hungarian algorithm. The reason is that the immediate task allocation from Hungarian algorithm usually flips many times and drives every robot to move back and forth. Moreover, the variance of finishing time using Hungarian algorithm is larger, meaning that the finishing time is, to some extent, based on luck, *i.e.*, the "random" allocation flips sometimes drive the robots to quickly approach a closer unassigned task locations, which can fortuitously reduce the time once the localization has converged; naturally the random flips also contribute to increase time. Besides, Table 1 shows that assignments using Interval Hungarian algorithm are more likely to assign the tasks that are consistent to the true scenarios, while only needing to reduce the uncertainty by improving the localization of the uncertainty-robots (other localized-robots simply wait). In this way, much overall system energy can be saved.



(a)



(b)

Figure 12: Assignment flips and finishing time during task executions

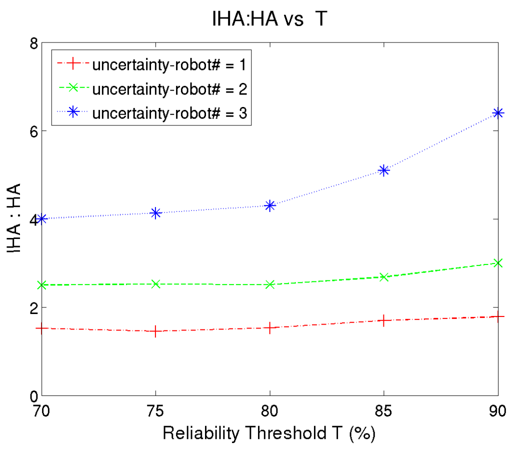


Figure 13: Success ratios with different reliability threshold T.

We also tested the impact of reliability threshold T on the task accomplishment quality. Since we have shown that the difference between simulation and physical experiments is nugatory, we carried out the evaluation of T in simulation. The three curves in Figure 13 show the results with 1, 2 and 3 uncertainty robots, respectively. The horizontal axis denotes the reliability threshold, and the vertical axis represents the success ratio *IHA*:*HA*, as defined before. From Figure 13 we can conclude: (1) the larger the threshold T, the greater the chance of success, and (2) more uncertainty-robots imply a larger success ratio for a given reliability threshold. The reason for (1) is that a larger T means a higher reliability level will eventually yield an assignment result consistent with current optimal solution. The reason for (2) is that more uncertainty-robots result in greater uncertainty, and less chance of success, while our algorithm's pessimism improves the successful chance by requiring each uncertainty-robot to meet a certain reliability threshold. However, larger T naturally entails more work being performed in obtaining fine utility estimations, and should be restricted by practical considerations for the application at hand. (Actually, in our case when $T \geq 90\%$, each robot is almost required to be completely localized).

Figure 14 shows the running time of Hungarian algorithm and Interval Hungarian algorithm across different assignment sizes. Both algorithms were implemented using C++ with the Standard Template Library for the data structures. The running times are for a desktop class machine (4GB memory and 2.8GHZ dual-core CPU). The matrices we used are square matrices, and the time of Interval Hungarian algorithm is only for one interval, since interval computations are independent and distributed across the robots. The results shows that for assignments with size fewer than 200, the uncertainty measurements cost only three seconds. We also believe that the practical running time could be greatly improved given more efficient data structures and better programming techniques.

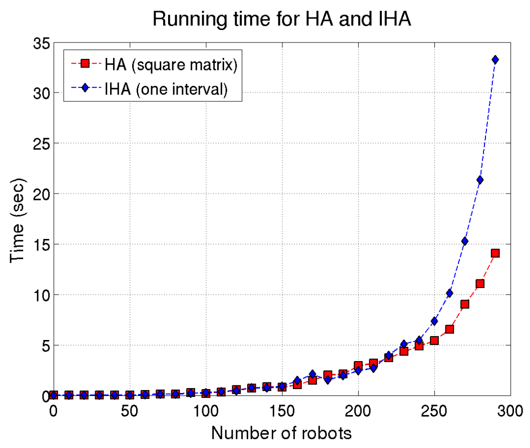


Figure 14: Running time for Hungarian algorithm and Interval Hungarian algorithm.

8 Conclusion

This paper presents the Interval Hungarian algorithm and an approach for measuring the effect of uncertainty on the optimality and stability of the output from the optimal assignment problems with direct applicability to task-allocation in multi-robot systems for a range of multi-robot applications. The Interval Hungarian algorithm is based on the bipartite matching variant of the Hungarian algorithm. Given an input utility matrix it outputs an assignment matrix along with intervals which are associated with each utility in the input. Each interval conveys a tolerance of the optimal assignment to perturbations in the associated utility value. We illustrated how uncertainties in multi-robot assignment problems can be quantified when probability distributions describe the utility estimates. Data from simulated and physical robot experiments were presented and compared and the results illustrate the advantages of our algorithm.

In broader applications, we believe the Interval Hungarian algorithm can be applied to any assignment problem that possesses uncertainties. The thresholding technique we describe is merely one example method for assessing the effect of uncertainty, although it appears to work well for Gaussian-distributed utilities. However, the process of designing other measurement methods (including for less common utility distributions) remains heuristic. Future work should consider principled approaches for this aspect of the problem.

References

[Atay and Bayazit, 2006] Nuzhet Atay and Burchan Bayazit. Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem.

Technical Report WUCSE-2006-54, Department of Computer Science & Engineering, Washington University, 2006.

[Berhault *et al.*, 2003] Marc Berhault, He Huang, Pinar Keskinocak, Sven Koenig, Wedad Elmaghraby, Paul Griffin, and Anton J. Kleywegt. Robot Exploration with Combinatorial Auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, pages 1957–1962, Las Vegas, NV, U.S.A., October 2003.

[Bernstein *et al.*, 2000] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov Decision Processes. In *Conference on Uncertainty in Artificial Intelligence*, June 2000.

[Bertsekas, 1990] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 1990.

[Dias and Stentz, 2001] M Bernardine Dias and Anthony (Tony) Stentz. A market approach to multirobot coordination. Technical Report CMU-RI -TR-01-26, Robotics Institute, Pittsburgh, PA, August 2001.

[Dias *et al.*, 2006] M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE — Special Issue on Multi-robot Systems*, 94(7):1257–1270, July 2006.

[Dinkelbach, 1969] W. Dinkelbach. *Sensitivitatsanalysen und Parametrische Programmierung*. Springer, Berlin, Heidelberg, New York, 1969.

[Fox, 2001] Dieter Fox. KLD-sampling: Adaptive particle filters. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS-14)*, pages 713–720, Cambridge, MA, U.S.A., 2001. MIT Press.

[Gerkey and Matarić, 2002] Brian P. Gerkey and Maja J. Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation — Special Issue on Multi-robot Systems*, 18(5):758–768, October 2002.

[Gerkey and Matarić, 2004] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, September 2004.

[Gerkey *et al.*, 2003] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. *Proceedings of the 11th International Conference on Advanced Robotics*, June 2003.

[Guestrin *et al.*, 2002] C. Guestrin, S. Venkataraman, and D. Koller. Context-Specific Multiagent Coordination and Planning with Factored MDPs. In *Proceedings of the eighteenth AAAI National Conference on Artificial Intelligence (AAAI'02)*, pages 253–259, Edmonton, Alberta, Canada, July 2002.

[Kuhn, 1955] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 1955.

[Lovász and Plummer, 1986] László Lovász and Michael D. Plummer. *Matching Theory*. North-Holland, 1986.

[Mills-Tettey *et al.*, 2007] G. Ayorkor Mills-Tettey, Anthony (Tony) Stentz, and M Bernardine Dias. The dynamic hungarian algorithm for the assignment problem with changing costs. Technical Report CMU-RI-TR-07-27, Robotics Institute, Pittsburgh, PA, July 2007.

[Munkres, 1957] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Soc. for Industrial and Applied Math.*, March 1957.

[Parker, 1998] Lynne E. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[Pentico, 2007] David W. Pentico. Assignment problems: A golden anniversary survey. *European J. of Op. Research*, 176(2):774–793, 2007.

[Roth *et al.*, 2006] M. Roth, R. Simmons, and M. Veloso. What to Communicate? Execution-Time Decision in Multi-agent POMDPs. In *Proceedings of the eighth International Conference on Distributed Autonomous Robotic Systems (DARS'06)*, pages 177–186, Minneapolis, MN, U.S.A., July 2006.

[Toroslu and Üçoluk, 2007] Ismail H. Toroslu and Göktürk Üçoluk. Incremental assignment problem. *Information Sciences*, March 2007.

[Ulrich and Borenstein, 1998] Iwan Ulrich and Johann Borenstein. VFH+: reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 1572–1577, Leuven, Belgium, May 1998.

[Vaughan and Gerkey, 2006] Richard T. Vaughan and Brian Gerkey. Really Reused Robot Code from the Player/Stage Project. In Davide Brugali, editor, *Software Engineering for Experimental Robotics*, Springer Tracts on Advanced Robotics. Springer, Berlin, Germany, 2006.

Linear Programming-based Sensitivity Analysis

The Linear Optimal Assignment Problem can be expressed as the following Integer Linear Program in n^2 variables:

$$\textbf{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{n} u_{ij} x_{ij}$$

$$\textbf{Subject to } x_{ij} \in \{0,1\},$$
$$\sum_{i=1}^{n} x_{ik} \leq 1, \quad k \in \{1 \cdots, n\},$$
$$\sum_{j=1}^{n} x_{lj} \leq 1, \quad l \in \{1 \cdots, n\}.$$

It is well known that first (integral values) constraint can be dropped [Pentico, 2007]: the problem is equivalent to the relaxed linear program because the constraint matrix is totally unimodular.

An intuitive instance of sensitivity analysis for a linear program can be understood by manipulating the right-hand sides of constraints. Adding (or subtracting) a small ϵ to the constant on the right-hand side shifts the polytope which defines the feasible region. Duality allows one to consider the linear program transformed so that the u_{ij} values of the input utility matrix now form exactly these right-hand side coefficients. The interpretation becomes clearer if one expresses this dual program in augmented form, involving only equality constraints and non-negative slack variables. Once the program is solved, the values of these slack variables denote the maximum deviation one of the right-hand side coefficients (*i.e.*, values of u_{ij}) may take before the solution involved a different basis — that is, results in a different assignment.

One key advantage of the introduced Interval Hungarian Algorithm over this indirect slack variable analysis is its efficiency. Although linear programming has a polynomial run-time, the $\sim O\big((n^2)^3\big)$ is significantly worse than the $O(n^4)$ (or $O(n^3)$ parallelized variety) presented herein.