Increasing allocated tasks with a time minimization algorithm for a search and rescue scenario

This item was submitted to Loughborough University's Institutional Repository by the/an author.


Additional Information:

- © 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

Metadata Record: https://dspace.lboro.ac.uk/2134/18846

Version: Accepted for publication

Publisher: © IEEE

Please cite the published version.
Increasing Allocated Tasks with a Time Minimization Algorithm for a Search and Rescue Scenario

Joanna Turner, Qinggang Meng and Gerald Schaefer

Abstract—Rescue missions require both speed to meet strict time constraints and maximum use of resources. This study presents a Task Swap Allocation (TSA) algorithm that increases vehicle allocation with respect to the state-of-the-art consensus-based bundle algorithm and one of its extensions, while meeting time constraints. The novel idea is to enable an online reconfiguration of task allocation among distributed and networked vehicles. The proposed strategy reallocates tasks among vehicles to create feasible spaces for unallocated tasks, thereby optimizing the total number of allocated tasks. The algorithm is shown to be efficient with respect to previous methods because changes are made to a task list only once a suitable space in a schedule has been identified. Furthermore, the proposed TSA can be employed as an extension for other distributed task allocation algorithms with similar constraints to improve performance by escaping local optima and by reacting to dynamic environments.

I. INTRODUCTION

Recently, there has been an increasing focus on developing multi-vehicle systems to complete jobs and missions in various fields including search and rescue missions, space and underwater exploration [1], [2]. Using multiple, specialized vehicles in these areas entails several advantages over one single all-purpose vehicle. They can be more cost-effective, are able to cover more ground, and are more resilient to failures [3].

The challenge however is to co-ordinate multiple vehicles to perform tasks while optimizing a chosen objective [4]. Considering a search and rescue scenario, where survivors need to be rescued before specified deadlines, the two main objectives are to maximize the number of survivors while minimizing the average waiting time before their rescue. One option is to use a centralized system, where a central server gathers information from each vehicle in the team, and then computes an allocation for each vehicle. The benefit of this approach is that the central server can optimize a chosen global objective based on a complete set of information from all vehicles. The drawbacks are the resulting single point of failure, and the requirement that each vehicle must have a communication link with the central server. Thus, the possible mission range is limited, and a heavy communication and computation burden is put on the central server. Distributed methods for task allocation can be used to overcome these limitations. Here, the planner runs on each vehicle simultaneously and the solution is reached through the interaction and exchange of information between them [3].

In the case of multi-assignments, where each vehicle can be allocated a sequence of one or more tasks, the complexity of the task allocation problem becomes NP-hard [3]. Consequently, as the number of tasks and vehicles increases, it is usually too computationally expensive to consider each possible combination of tasks for each possible vehicle in order to find the optimal solution. Therefore, heuristic methods are employed to speed up the process of finding a solution while maintaining an efficient and scalable algorithm [5].

The consensus-based bundle algorithm (CBBA) [6] was developed to tackle the distributed multi-assignment task allocation problem. It employs a greedy auction strategy to generate a task bundle, followed by a consensus procedure to resolve any conflicting assignments arising from inconsistencies in situational awareness. Of the various extensions and modification, [7] and [8] address multi-agent task assignments and heterogeneous networks. [9] incorporates time windows of validity and fuel costs as part of the scoring scheme as well as real-time re-planning for broken communication links. [10] extends the CBBA to permit asynchronous communication, while [11] developed a strategy based on the CBBA and the algorithm from [12], for assigning critical tasks where agents have limited capacity. [13] propose an alternative method using a game theory approach for resolving conflicts. Most relevant to this paper is the approach in [14], which introduces the concept of a significance value of a task. Rather than using a greedy approach to task inclusion, the strategy there aims to directly optimize a global mathematical objective. Specifically, it addresses the minimization of the average waiting time of survivors in a search and rescue scenario with deadlines. In this case, the significance of a task is measured as its start time in a vehicle’s schedule with the addition of delays caused to later tasks. Vehicles will include into their schedules the tasks whose significance they can improve the most.

This paper proposes the Task Swap Allocation (TSA) algorithm extension of the Significance Method (SM) in [14]. It is designed to iteratively improve the total number of allocated tasks after the original solution is generated. While the SM aims to optimize waiting time and meet deadlines, it does not effectively prioritise the maximisation of task assignments with regards to those deadlines. This results in potentially feasible task assignments being rendered infeasible due to the ordering and allocation produced by the SM. With the aim of
maximizing survivors rescued, the proposed TSA redefines the global objective from minimizing average waiting time to maximizing tasks allocated. It takes advantage of existing space in a vehicle’s schedule to shift assigned tasks among vehicles in order to create a feasible space for unassigned tasks. While the paper focuses on a search and rescue scenario, the algorithm can be applied to other scenarios, including cleaning chemical spills, patrolling, and checking for structural integrity.

II. PROBLEM STATEMENT

Consider a search and rescue scenario with \( n \) heterogeneous autonomous vehicles and \( m \) survivors. The goal is to provide targeted emergency support to the survivors as quickly as possible. Some may require food supplies, while others may require medical provisions. Different types of vehicles will be capable of executing different types of tasks to match specific needs. With the knowledge of survivors’ locations, and a planner on each vehicle, the vehicles will use local communication to co-ordinate a rescue plan to provide the necessary support. For this problem, each survivor must be visited by at least one vehicle in order to be deemed rescued. Each vehicle will sequentially visit the targets assigned to it, but is not required to return to its initial location. The main challenge is assigning the tasks in a way that satisfies the global objectives. While aiming to maximize the number of survivors rescued by reaching them before their individual deadlines, waiting times should also be minimized since health conditions are time critical. A deadline reflects an assumed given estimate of the remaining time available to successfully reach a survivor before they are considered “unrescuable”. In this scenario, the term task and survivor are used interchangeably.

To formulate the problem mathematically, \( \mathbf{V} = [v_1, \ldots, v_n]^T \) denotes the set of heterogeneous autonomous vehicles and \( \mathbf{T} = [t_1, \ldots, t_m]^T \) the set of survivors waiting to be rescued. The allocation \( \mathbf{A} = [a_1, \ldots, a_n]^T \) is a partition of \( \mathbf{T} \), where \( a_i, i = 1, \ldots, n \), is an ordered list of survivors allocated to be rescued by vehicle \( v_i \). The optimization objective of minimizing waiting time is formulated as

\[
J = \min \left\{ \frac{1}{m} \sum_{j=1}^{n} \alpha_i k(a_i) \right\},
\]

where \( c_{i,k}(a_i) \) represents the time cost associated with vehicle \( v_i \) rescuing the \( k \)-th survivor in \( a_i \), and \( \alpha_i \) represents the number of survivors assigned to \( v_i \). The constraints are as follows:

\[
|a_i| \leq m_i
\]

\[
\bigcup_{i=1}^{n} a_i = T, a_i \cap a_j = \emptyset \text{ with } i \neq j,
\]

compatibility matrix \( \mathbf{H} \) with entries \( h_{i,k} = 1 \) if the type of vehicle \( i \) is capable of performing the type of task \( k \). Equation (5) adds a constraint on the possible start time of a rescue. Each survivor has an individual deadline for starting a rescue \( s_{k} \). The start time \( s_{k,a} \) for rescuing the \( k \)-th survivor, determined by its position in a vehicle’s ordered task list, can therefore be no later than \( s_k \). A symmetric communication matrix \( \mathbf{G}(t) \) is also defined, where an entry \( g_{i,j}(t) = 1 \) indicates that a direct communication connection exists between vehicles \( v_i \) and \( v_j \) at time \( t \).

III. SIGNIFICANCE METHOD

This section describes the key aspects of the Significance Method as defined for the search and rescue scenario. This description will be helpful for describing the proposed TSA.

The SM is a distributed task allocation algorithm that aims to directly optimize a specified objective, using a concept called significance. This method differs from the CBBA, which uses an auction-based approach and therefore cannot directly target an objective to optimize. Inspired by the CBBA, the SM iterates over two phases, the task inclusion phase and the consensus phase. In the former, a vehicle will iteratively include tasks that provide the greatest improvement to the global objective. This improvement is determined by the greatest difference between a task assignment’s current global significance value, and the significance it would have in a vehicle’s task list who is not already assigned to that task. In the latter phase, conflicting assignments are resolved. These conflicts occur when two or more vehicles are assigned to the same task. In this instance, the vehicle with the lowest significance will keep the task, while the others release it through a task removal procedure. The whole process repeats until no further improvements can be made, and all conflicts are resolved.

A. Significance

The key idea is that the significance of a task, within a certain position in a vehicle’s task list, represents its local impact on the objective being optimized. Given a task \( t_k \) assigned to a vehicle \( v_i \), the definition of significance is

\[
w_{k}(a_i \oplus t_k) = c_i(a_i) - c_i(a_i \ominus t_k),
\]

where \( c_i(a_i) \) is the total cost incurred by vehicle \( v_i \) when visiting all survivors in \( a_i \), \( a_i \oplus t_k \) denotes the removal of task \( t_k \) from its position in the ordered list \( a_i \). If the task were to be reinserted in the same position in the task list from which it was removed, the value of its significance would be the same. It is bidirectional.

With the objective of minimizing waiting time, the cost of a task is the time at which a vehicle can start its execution.
The significance of a task then becomes its time cost, with the addition of any delays incurred to later tasks as a result of that task’s inclusion. Significance is then formally defined as

$$w_k(a_i \otimes t_k) = c_{i,b}(a_i) + \sum_{r=b+1}^{\infty} \{c_{i,r-1}(a_i) - c_{i,r-1}(a_i \otimes t_k)\}, k \in a^T_i,$$  

(7)

where \(b\) represents the position of task \(t_k\) in \(v_i\)’s task list, \(c_{i,b}(a_i)\) denotes the time cost of \(t_k\) at position \(b\) in \(v_i\)’s task list, and the summation computes the delays incurred to later tasks in the task list. Each vehicle stores a significance list to record every task’s significance value, defined as \(\gamma_i = [w_1, \ldots, w_m]^T\). A global consensus is reached when each vehicle has an identical copy of this list.

B. Task Inclusion Phase

To determine which task to include, a vehicle first computes the marginal significance for each task not currently in its task list. Like significance, marginal significance is also calculated as its start time and any delays caused to later tasks. It is computed by temporarily inserting the task into the vehicle’s task list at each available position. The value for the position incurring the lowest total for cost and delays is recorded as the task’s marginal significance. The term marginal here is used to distinguish it from significance. When a task is assigned to a vehicle, its value is defined as its significance. When a vehicle is determining which task to include, the value computed is known as marginal significance. A list to store the marginal significance on each vehicle is defined as \(\gamma_i^m = [w_1^m, \ldots, w_m^m]^T\). The vehicle will look to include the task that gives the greatest improvement to the global significance, satisfying

$$\max_{k=1}^m \{\gamma_{i,k}^m - \gamma_{i,k}^m\} > 0, k = 1, \ldots, m.$$  

(8)

The task corresponding to the maximum improvement towards the global objective is then included into the vehicle’s ordered task list. This phase repeats until no more tasks can be added.

C. Consensus and Task Removal Phase

Once the task inclusion phase is complete, the significance of all tasks and the vehicle IDs associated with those values are updated and broadcast to vehicles where \(g_{i,j} = 1\). A consensus procedure is then used to resolve conflicting assignments. Vehicles with a higher significance for a conflicting assignment will be the ones to release the task. Once significance and associated vehicle ID values have been updated following consensus, if a vehicle has several tasks to remove, they are iteratively removed in order of greatest objective improvement. When this phase is completed, the process starts again from the task inclusion phase, and repeats until no inclusions or removals can be made for a given length of time. At this point, the system is deemed to have converged and the task allocation process ends.

IV. THE TASK SWAP ALGORITHM

The proposed Task Swap Allocation (TSA) algorithm improves the Significance Method by increasing the number of allocated tasks when possible. The solution generated by the SM is often dependent on the order and combination of task inclusions. If a solution is reached that is suboptimal with regards to number of tasks allocated, there is no functionality to reallocate tasks to create a feasible space for unallocated tasks. In a search and rescue mission, this limitation would result in survivors not being rescued when they could have otherwise been.

A recurring situation where a maximal allocation exists but cannot be reached is described in the following. Consider the scenario shown in Fig. 1(a). To satisfy (5), a task’s start time must be before its deadline, although it may end after the deadline. The time delay before and between tasks represents the time required for vehicles to travel to the survivor locations. The delay in reaching the first task is dependent on a vehicle’s starting position. Tasks \(t_1\) and \(t_2\) have been allocated to minimize waiting time. If \(t_1\) and \(t_2\) were the only tasks, this allocation would be the optimal solution. Consider an unallocated task \(t_3\). Suppose that due to the locations of the vehicles and tasks, and \(t_3\)’s early deadline, \(v_1\) would be the only vehicle able to reach \(t_3\) in
time. However, as \( v_1 \) is currently assigned \( t_1 \), which also has an early deadline, \( v_1 \) cannot feasibly include \( t_3 \) into its task list. The solution, as illustrated in Fig. 1(b), would be for \( v_2 \) to take on \( t_1 \), making room for \( v_1 \) to include \( t_3 \). Although the waiting time for \( t_1 \) and \( t_2 \) has increased, all tasks would then be allocated achieving a maximum assignment. The SM is currently unable to perform such an exchange when optimizing waiting time.

To facilitate the exchange of tasks between vehicles with the aim of optimizing the number of allocations, the computation of significance is modified to meet the objective of maximizing task allocations. For this objective, significance reflects a task’s potential to make room for an unallocated task. This change required to make room for an unallocated task. All unallocated tasks are set to have the highest significance value, defined as \( U_{sig} \). Assigned tasks that could feasibly be replaced by an unallocated task then have a relatively lower significance than \( U_{sig} \) based on a reduction rate \( r \). For the remainder of this paper, \( r = 0.5 \). The significance successively decreases by \( r \) for each additional change required to make room for an unallocated task. This approach allows vehicles to release tasks that can make space for unallocated tasks after being reassigned.

Fig. 2 illustrates how the significance of a task encourages the least disruptions to existing task lists. Assuming \( v_3 \) could include the unallocated task \( t_4 \) in place of \( t_3 \), \( t_3 \)’s significance becomes half that of \( t_4 \): \( U_{sig} \times r = 100 \). \( v_2 \) can include \( t_3 \) in place of \( t_2 \), giving \( t_2 \) a significance of 50 in \( v_2 \)’s task list. \( v_1 \) is capable of including both \( t_3 \) and \( t_2 \). As \( t_3 \)’s significance is higher, \( v_1 \) prioritizes \( t_3 \) for inclusion thereby preserving \( v_2 \)’s task list from unnecessary changes. This is desirable as following the SM, \( t_2 \)’s task list will remain optimized for minimizing average waiting time. The significance of a task in the context of maximizing task allocations is formally defined as

\[
w_k(a_i \ominus t_k) = \max\{F_{i,k,u} \times r\}, k \in a_i^u, 0 < r < 1, \quad (9)
\]

where \( F \) is the list of significance values belonging to tasks that can feasibly fit into \( v_1 \)’s task list \( a_i \), when task \( t_k \) is removed. If a task in \( a_i \) can be swapped for two or more tasks with different significance values, the highest value is recorded.

Algorithm 1 describes the process of computing the significance of tasks in a vehicle \( v_i \)’s task list, for optimising allocated tasks. The algorithm runs following the Consensus, Task Removal and Task Inclusion phases. Task candidates include unallocated tasks and tasks assigned to other vehicles with a significance value higher than 0. A candidate task is considered to fit into a task list if each task’s deadline can be respected given its inclusion.

**Algorithm 1** Computing Task Swap significance values for tasks in \( v_i \)’s task list

1. TaskList = \( v_i \)’s task list
2. Set significance of tasks in TaskList to 0
3. TaskCandidates = tasks not in TaskList with a significance > 0
4. \( r = 0.5 \)
5. for each task \( L \) in TaskList do
   6. for each task \( C \) in TaskCandidates do
      7. if significance(\( C \) * \( r \) > significance(\( L \)) then
         8. TempTaskList = TaskList
         9. Remove task \( L \) from TempTaskList
         10. Adjust Task Times in TempTaskList
         11. for each position \( p \) in TempTaskList do
             12. if \( C \) fits in TempTaskList at \( p \) then
                13. significance(\( L \)) = \( r \) * significance(\( C \))
                14. break
             end if
         end for
      end if
   end for
end for

A maximum number of swaps, expressed as “swap distance” \( SD_{max} \) and designed to limit the number of moves in a swap sequence and thus the number of iterations and communications, can be defined to limit the number of changes allowed to make space for an unallocated task. A lower limit

\[
LL = U_{sig} \times r^{SD_{max}}, 0 < r < 1. \quad (10)
\]

is applied to the significance of a task. If the significance falls below this limit, it is set to zero, thereby preventing it from being considered for reallocation.

The proposed TSA using the significance defined in Equation (9) runs after the SM solution is reached. The TSA steps illustrated by the flowchart in Fig. 3 follow those of the SM with two adjustments. Tasks considered for inclusion are those with a significance value greater than zero. If such a task can be included in the vehicle’s current task list, the task’s marginal significance is set to zero. This ensures that the task with the greatest significance is selected for inclusion, thereby minimizing the number of changes as
depicted in Fig. 2. Additionally, the number of tasks removed during the task removal phase is limited to one. A vehicle may have two or more tasks in its list able to swap with the same unallocated task. If one is removed to make space, the others need not be removed. If the two tasks are able to swap for different unallocated tasks, they will be removed in the following rounds. This strategy further preserves the positive synergies attained with the SM solution. The three steps that differ in the proposed approach from the SM are highlighted in Fig. 3.

Fig. 4 illustrates the effect of the proposed TSA algorithm as an extended scenario of that shown in Fig. 1. Fig. 4(a) shows a solution following the time optimization SM with one unallocated task \( t_4 \). Given the locations and time restrictions, the only feasible solution with all tasks allocated is determined to be that shown in Fig. 4(c). The steps to reach it through reallocation are illustrated in

---

**Fig. 3:** Flowchart of the proposed TSA running at vehicle \( v_1 \). The highlighted steps are those that differ compared to the SM algorithm. The principle difference is that in TSA significance is computed to maximize task allocations, while SM computes significance to optimize average waiting time.

**Fig. 4:** Vehicles’ schedules generated by SM. \( t_4 \) is unallocated. A task must be started before the deadline in order to rescue that survivor. (a) \( t_1, t_2 \) and \( t_3 \) are optimized to minimize waiting time but \( t_4 \) is unallocated. \( v_1 \) is the only vehicle close enough to reach \( t_4 \) in time, but cannot feasibly include it into its schedule given \( t_1 \). (b) The reallocation steps taken with the proposed TSA to create a feasible space for \( t_4 \). \( v_3 \) first includes \( t_2 \), moving back \( t_3 \)’s start time. \( v_2 \) then includes \( t_1 \), allowing space for \( v_1 \) to include \( t_4 \). (c) The maximal allocation solution generated by the proposed TSA.

---

**V. NUMERICAL RESULTS**

This section presents the results of numerical simulations conducted to test the performance of the proposed TSA. The improvement in terms of number of tasks allocated compared with the SM is assessed, followed by a comparison with CBBA and SM in terms of average waiting time. Finally an overall analysis of its average performance is given.

**A. Scenario and Simulation Setup**

To test the system, a rescue team equally split into two vehicle types is considered. The first provides medicine, while the second supplies food. Their speed is assumed to
be constant and is set to 30m/s and 50m/s respectively. The survivors are likewise equally split into those requiring food and those requiring medicine. The medicine tasks last for a duration of 300 seconds, while the food tasks last 350 seconds. The deadlines for starting each rescue are uniformly randomly distributed on a timeline between 0 and 2000 seconds. The mission takes place in a 3D space spanning 10000m x 10000m x 1000m. The tasks are uniformly positioned within this 3D space, while the vehicles’ starting positions are uniformly placed on the 2D ground space. A row formation is used for these experiments, where each vehicle has a constant communication link with exactly one other vehicle.

### B. Simulation Results

Ten different scenarios were tested involving a different number of tasks and vehicles, each conducted 50 times. The numbers of vehicles tested were 6, 8, 10, 12 and 14. Two task-to-vehicle ratios, 4.6 and 2, were tested to assess the performance of the system under high and medium demand respectively. The 4.6 task-to-vehicle ratio was experimentally set to test the system under high demand, i.e., approaching maximum capacity. It is worth noting that given the random initialization of task and vehicle locations and deadlines, it is sometimes impossible for some tasks to be started by any vehicle before its deadline.

The obtained results are given in Table I which shows, for each scenario, the percentage of the 50 simulations in which the proposed TSA altered the solution generated by the SM, as well as the best, mean, and worst percentage changes to the number of tasks allocated when reallocations occurred. For a ratio of 2, only a minority of solutions were improved, with the 6 vehicles 12 tasks scenario only allocating one extra task overall. This indicates that when there are fewer tasks per vehicle, there is less room for improvement as the SM is more likely to find a maximal allocation solution. The difference between the best, mean and worst percentage change is relatively small. For the ratio of 4.6, the majority of solutions were improved by the proposed TSA. As the number of tasks and vehicles increases, the proportion becomes greater with the last scenario having nearly all solutions improved. The greatest difference was a 20% increase in survivors rescued for the 10 vehicles 46 tasks scenario. In this case, 7 extra survivors were rescued as a result of the proposed TSA.

The graphs illustrate a steady rate of increase in average time cost between the three algorithms. In the third example, SM allocates 3 more tasks than CBBA, and TSA allocates an additional 3 tasks. The increase in average time is approximately 40s and 42.7s per task respectively.

Table II shows the average performance of the TSA over 50 simulations compared with the SM and the CBBA for each of the ten scenarios with regards to task allocations and iterations. The total of iterations for one simulation is determined by the last time an allocation change was made, either through inclusion or removal. As expected, TSA provides the highest average allocation in all scenarios. The improvement is marginal compared to the difference between CBBA and SM. In part, this is because the TSA does not make allocation changes in every instance; its average is determined by the last time an allocation change was made, either through inclusion or removal. As expected, TSA provides the highest average allocation in all scenarios. The improvement is marginal compared to the difference between CBBA and SM.

Performance. Performances are analyzed by vehicle type. To appreciate the gain in performance of TSA, the analysis also includes the performance of the CBBA algorithm. For these results, an additional round of the SM was added following the proposed TSA to optimize the time cost of tasks added during the TSA. The graphs illustrate a steady rate of increase in average time cost between the three algorithms. In the third example, SM allocates 3 more tasks than CBBA, and TSA allocates an additional 3 tasks. The increase in average time is approximately 40s and 42.7s per task respectively.

![Figure 5](image)

**Table I: TSA’s Performance Measured as Percentage Change in Task Allocations over 50 Simulations. Shown are the Percentage of Simulations Where the Proposed Method Changed the Task Allocation Solution Generated by the SM Algorithm, and the Best, Mean, and Worst Percentage Change to the Total Number of Allocated Tasks.**

<table>
<thead>
<tr>
<th>Ratio</th>
<th>Scenario</th>
<th>Modified</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>v6 t12</td>
<td>2%</td>
<td>+9.09%</td>
<td>+9.09%</td>
<td>+9.09%</td>
</tr>
<tr>
<td></td>
<td>v8 t16</td>
<td>8%</td>
<td>+7.69%</td>
<td>+6.97%</td>
<td>+6.67%</td>
</tr>
<tr>
<td></td>
<td>v10 t20</td>
<td>12%</td>
<td>+5.88%</td>
<td>+5.72%</td>
<td>+5.26%</td>
</tr>
<tr>
<td></td>
<td>v12 t24</td>
<td>14%</td>
<td>+5.00%</td>
<td>+4.53%</td>
<td>+4.35%</td>
</tr>
<tr>
<td></td>
<td>v14 t28</td>
<td>8%</td>
<td>+4.00%</td>
<td>+3.85%</td>
<td>+3.96%</td>
</tr>
<tr>
<td>1/4.6</td>
<td>v6 t28</td>
<td>74%</td>
<td>+14.29%</td>
<td>+6.45%</td>
<td>+3.85%</td>
</tr>
<tr>
<td></td>
<td>v8 t36</td>
<td>78%</td>
<td>+14.81%</td>
<td>+6.04%</td>
<td>+3.03%</td>
</tr>
<tr>
<td></td>
<td>v10 t46</td>
<td>84%</td>
<td>+20.00%</td>
<td>+4.88%</td>
<td>+2.38%</td>
</tr>
<tr>
<td></td>
<td>v12 t56</td>
<td>88%</td>
<td>+11.30%</td>
<td>+4.69%</td>
<td>+1.92%</td>
</tr>
<tr>
<td></td>
<td>v14 t64</td>
<td>94%</td>
<td>+10.42%</td>
<td>+4.78%</td>
<td>+1.67%</td>
</tr>
</tbody>
</table>

**Table II: Average Task Allocations and Iterations Performance of TSA Compared with SM and CBBA over 50 Simulations.**

<table>
<thead>
<tr>
<th>Ratio</th>
<th>Scenario</th>
<th>Task Allocations (Avg)</th>
<th>SM w/ TSA</th>
<th>Iterations (Avg)</th>
<th>TSA (only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2</td>
<td>v6 t12</td>
<td>10.40</td>
<td>11.38</td>
<td>11.40</td>
<td>3.82</td>
</tr>
<tr>
<td></td>
<td>v8 t16</td>
<td>13.80</td>
<td>15.22</td>
<td>15.30</td>
<td>5.58</td>
</tr>
<tr>
<td></td>
<td>v10 t20</td>
<td>17.36</td>
<td>19.22</td>
<td>19.34</td>
<td>7.26</td>
</tr>
<tr>
<td></td>
<td>v12 t24</td>
<td>20.92</td>
<td>22.92</td>
<td>23.06</td>
<td>8.56</td>
</tr>
<tr>
<td></td>
<td>v14 t28</td>
<td>24.44</td>
<td>26.86</td>
<td>26.94</td>
<td>12.08</td>
</tr>
<tr>
<td>1/4.6</td>
<td>v6 t28</td>
<td>19.28</td>
<td>21.92</td>
<td>22.96</td>
<td>4.82</td>
</tr>
<tr>
<td></td>
<td>v8 t36</td>
<td>25.18</td>
<td>29.48</td>
<td>30.80</td>
<td>6.86</td>
</tr>
<tr>
<td></td>
<td>v10 t46</td>
<td>32.32</td>
<td>38.22</td>
<td>39.76</td>
<td>10.20</td>
</tr>
<tr>
<td></td>
<td>v12 t56</td>
<td>39.12</td>
<td>46.68</td>
<td>48.60</td>
<td>13.30</td>
</tr>
<tr>
<td></td>
<td>v14 t64</td>
<td>45.26</td>
<td>54.32</td>
<td>56.80</td>
<td>15.62</td>
</tr>
</tbody>
</table>
solutions that cannot be improved. For the average iterations comparison, the proposed TSA is considered in isolation as opposed to combined with the SM. The results show that for the 2 ratio, the small proportional change in allocations is accompanied by a similarly small average number of iterations. For the 4.6 ratio, the TSA consistently runs for approximately half the cycles needed by the SM.

VI. CONCLUSIONS

In a search and rescue mission, optimal task allocation for available vehicles is crucial. In this paper, an effective algorithm that allows for easy and efficient swap of allocated tasks is proposed to improve a previous method for task allocation. The employed strategy allows vehicles to re-allocate tasks to create a feasible space for unallocated tasks by taking advantage of existing schedule space. Distributed vehicles can negotiate task allocations by exchanging a single significance value. Simulations showed a noteworthy increase in performance, measured as the total number of survivors rescued, making the method appealing when this objective is a priority. A marginal increment in the number of iterations appeared proportionate to the gain in performance. This efficiency derives from the fact that changes are made to a task list only once a suitable space in a schedule has been identified. Experimental results confirmed that the proposed algorithm can beneficially be applied to an existing scheduling method [14], thus opening the possibility of integration to other implementations.

ACKNOWLEDGMENT

This work was supported by EPSRC (grant number EP/J011525/1) with BAE Systems as the leading industrial partner.

REFERENCES