INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL
Int. J. Robust. Nonlinear Control 2011; **21**:1358–1371
Published online 4 May 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/rnc.1717

Comparison of Tabu/2-opt heuristic and optimal tree search method for assignment problems

J. Jackson*, †, M. Faied and A. Girard

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, U.S.A.

SUMMARY

A nonlinear cooperative control problem involving several vehicles is detailed and solved. The vehicles must be assigned to perform many tasks such that they obey constraints on the order of task completion and minimize a nonlinear objective function, the total time to finish all tasks. This is an example of a combinatorial task assignment problem. A novel heuristic is introduced that represents a new combination of two combinatorial optimization tools. The quality of the solutions produced by this heuristic is demonstrated through comparison with a branch and bound search method. The branch and bound method is a well-known procedure and finds optimal solutions to the constrained, nonlinear task assignment problem. Copyright © 2011 John Wiley & Sons, Ltd.

Received 17 September 2010; Revised 19 January 2011; Accepted 19 January 2011

KEY WORDS: cooperative control; task assignment; combinatorial optimization; heuristic

1. INTRODUCTION

Many applications require the scheduling and assignment of multiple agents for the completion of tasks, e.g. manufacturing tasks, the pick up and delivery of packages or people, or the time critical scheduling of cooperative military missions. When the agents are mobile, this type of application is referred to as a vehicle routing problem (VRP). Often, the tasks have an associated precedence order such as the constraint that a package be picked up before it is delivered or the necessity to classify a target of interest before it is attacked. Many different task assignment algorithms have been developed, implemented, and simulated [1–11]. Owing to the complexity of the problem, and the practical necessity to solve it in near real time, most of the solution algorithms have been based on heuristic methods. In order to judge their effectiveness, it is desirable to compare them against an algorithm that produces an optimal solution to the given assignment problem. This work discusses a new and effective heuristic used to solve a practical variant of the VRP [12]. This method is compared with one that is provably optimal [13]. The VRP is specifically treated here, but the heuristic solution procedure is quite general and can be used to solve any assignment and ordering problem fitting the form used in the problem statement of (7)–(10). The optimal method uses an exhaustive search through all possible solutions to an assignment problem. It is general and can solve any deterministic variation of the VRP that can be represented by a search tree. The heuristic method combines the 2-opt [14] and Tabu search [15] heuristics and uses a decomposition of the problem in order to take advantage of the strengths of both heuristics. This method is able to contend with nonlinear objective functions and scales polynomially with the number of agents and tasks. There is no known analytical bound for the optimality gap for this metaheuristic applied

*Correspondence to: J. Jackson, Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, U.S.A.
†E-mail: justinpjack@gmail.com

to this problem, but an experimental optimality gap is obtained by comparison with optimality on small problems.

Task assignment problems appear in many domains including manufacturing, aircraft control and scheduling, military missions, and computer time scheduling. Owing to the complex nature of scheduling problems, the variety of problem structures, and needs related to implementation, many unique and creative methods have been used to solve these problems. The basic travelling salesman problem (TSP) is the canonical VRP. One salesman or vehicle is to visit several locations and return to the point of origin while minimizing the distance travelled. While the statement of this problem is simple, the search for better solution methods has been the source of much advancement in combinatorial optimization. When solved to optimality, this problem scales exponentially with the number of locations to be visited. There are also heuristics available to solve the problem approximately [14]. The Lin–Kernighan variable k-opt exchange heuristic [16] has been effectively implemented by Helsgaun [17]. This implementation of the Lin–Kernighan heuristic holds the record on solution cost for all TSP problem instances in the TSPLIB, a collection of standard test instances for TSP solution algorithms.

One important variation of the TSP is the multiple travelling salesman problem (mTSP). This variation is concerned with multiple salesmen tasked with visiting several locations. This problem has been addressed in the literature through linear programming formulations [18, 19] and through the use of transformations to the single TSP. This problem has been posed as the single or multiple depot mTSP, where the multiple depot mTSP has the vehicles beginning from and returning to separate depots. The sequential ordering problem is treated by Ascheuer *et al.* [20] and is concerned with satisfying precedence constraints dictating the ordering of certain tasks. The single TSP has been treated where the total length of the route is penalized in the cost function. Precedence constraints in the mTSP must be obeyed even when separate agents perform tasks that have a precedence relation defined between them.

Other methods used for vehicle routing use capacitated transshipment assignment formulations [21]. These methods are not computationally intensive and are scalable, but involve very little scheduling of the agent routes. The mixed integer linear programming formulation has been used to describe several types of VRPs [21] and is a versatile tool, but solution time scales exponentially with the number of tasks and agents. The branch and bound method has been used to solve complex VRPs exactly [22]. Such representations can involve complex cost functions and environmental factors such as wind and adversarial action.

Metaheuristics such as Tabu Search [15], Simulated Annealing [23], and Genetic Algorithms [24] have been used to solve VRPs. These metaheuristics are most useful when planning in the presence of many types of constraints that restrict the allowable order of task completion and when evaluating solutions with complicated, nonlinear cost functions. The Tabu search heuristic has been used to solve vehicle routing [25], flow shop scheduling [26], and generalized assignment problems [27]. These heuristics have the advantage of being able to treat large classes of problems, but have the disadvantage of minimal or no guarantee on the solution quality.

This paper presents several original contributions. Our solution representation allows the assignment and order constraints to be satisfied independently. The assignments and completion order remain coupled through the objective function, but our method allows feasible solutions to be computed in low-order polynomial-time. This separation also allows for different optimization methods to be used where appropriate. To our knowledge, the Tabu search has never been used with any variant of the k-opt repair heuristic. The problem of finding feasible vehicle assignments and the problem of finding a feasible task ordering are treated separately, which allows the use of this combined heuristic.

2. TABU/2-OPT ALGORITHM DEVELOPMENT

Consider an assignment problem where N_a agents are to be assigned to complete N_t tasks. The agents might be vehicles, personnel, or machines to be scheduled. The tasks can be a picture to

DOI: 10.1002/rnc

be taken, an item to be moved or assembled, or a weapon to be deployed. There are assignment constraints that restrict the possible assignments of agents to tasks and completion order constraints that restrict the possible order in time when tasks can be performed. The sets of available agents and tasks are $\mathscr{A}$ and $\mathscr{T}$, respectively. The available agents are,

$$a_i \in \mathscr{A}, \quad i = 1, \ldots, N_a, \tag{1}$$

and

$$T_j \in \mathscr{T}, \quad j = 1, \ldots, N_t, \tag{2}$$

are the tasks to be completed by the supplied agents. The notions of precedent and dependent tasks are used here to denote the set of all tasks that must occur earlier in time or later in time, respectively, than a given task. The occurrence time of a task T_j is t_j. The precedent and dependent sets are,

$$p_j = \{T_k \in \mathscr{T} \mid t_k < t_j\}, \quad j = 1, \ldots, N_t, \tag{3}$$

$$d_j = \{T_k \in \mathscr{T} \mid t_k > t_j\}, \quad j = 1, \ldots, N_t, \tag{4}$$

and capture all transitivity relationships. For this version of the VRP, the set p_j will be empty for at least one task and the set d_j will be empty for at least one task. The set p_j is used with d_j to quickly determine the feasibility of a given problem instance, this is addressed in (14). The sets $\mathscr{A}_{f_j}$ and $\mathscr{A}_{a_j}$, $j = 1, \ldots, N_t$, are the set of agents allowed to perform task T_j and the set of agents that are assigned to perform task T_j, respectively. Let X specify the assignments of vehicles to tasks and the order in which tasks are completed. A solution to the task assignment problem is given in the form of the set of agents assigned to complete each task $\mathscr{T}a$ and a partially ordered set giving the order in which the tasks should be completed $\mathscr{C}o$. That is $X = \{\mathscr{T}a, \mathscr{C}o\}$, where

$$\mathscr{T}a = \{\mathscr{A}_{a_1}, \ldots, \mathscr{A}_{a_{N_t}}\}, \tag{5}$$

$$\mathscr{C}o = \{u_1, \ldots, u_{N_t}\}. \tag{6}$$

Here, u_i represents the task that is placed at position i in the partial ordering. The arrival time of an agent a_i at a task T_k is denoted as $t_{a_{ik}}$. The arrival time of agent a_i at its final location $t_{a_{ie}}(X)$ is a function of this specification. The optimization problem is,

$$\min_X \left\{ \max_i \{t_{a_{ie}}(X)\} \right\} \quad \text{s.t.} \tag{7}$$

$$t_k > t_j \quad \forall T_j \in p_k, \quad k = 1, \ldots, N_t, \tag{8}$$

$$t_k > t_{a_{ik}} \quad \forall a_i \in \mathscr{A}_{a_k}, \quad k = 1, \ldots, N_t, \tag{9}$$

$$\mathscr{A}_{a_j} \subseteq \mathscr{A}_{f_j}, \quad j = 1, \ldots, N_t. \tag{10}$$

The objective here is to minimize the mission time (7), that is, the time for the last agent to complete its final task. This objective function is a function of the assignments of agents to tasks, the order of completion of the tasks, and the vehicle kinematics. Note that only the set p_k is used in the problem specification, d_k is used solely to check whether the problem can be solved by this heuristic method (14). The operational constraints restrict agents to only perform tasks after all precedence constraints have been met (8) and after every agent involved in a task's completion has arrived at said task (9). These constraints also restrict the agents permitted to be assigned to a task (10). This problem is of particular interest because our research considers the planning of military missions [21]. The types of constraints considered occur naturally in such missions. It is also appropriate to minimize the total duration of the mission. This formulation is flexible enough to allow agents to cooperate to complete individual tasks if this helps minimize the cost.

The properties of the agents and the tasks that are assumed throughout this work are the following. For simplicity, the agents are assumed to be unconstrained by fuel limits or any other

DOI: 10.1002/rnc

endurance constraints. The agents are assumed to perform every task to which they are assigned successfully. Tasks are assumed to have zero completion time; only travel time is considered. The vehicles are assumed to obey steerable unicycle kinematics, operate in the Euclidean plane, and have no limits on turn-rate. The vehicle velocities are limited by $V_i \in [0, V_{\max_i}]$, where $V_{\max_i}$ is the maximum travel velocity of agent a_i. Endurance constraints and limits on resources cannot be directly addressed by this structure. Note that timing constraints are not explicitly incorporated in the completion order. Algorithm 1 guarantees that if the completion ordering satisfies order constraints, the resulting times of occurrence satisfy (8). If two tasks that have an order relationship between them are assigned to different agents, the timing must be enforced by the constraints (8).

To determine the cost of a feasible solution to the optimization problem, the following must be known: the agents that are assigned to all of the tasks, the order in which each of the agents will complete the tasks it is assigned to, and the kinematics it will follow to perform these tasks. The set $\mathscr{T}a$ represents the assignments for each task, the partially ordered set $\mathscr{C}o$ represents the order of the task completion. The optimization procedure is performed in two levels. The lower level performs the optimization of the completion order. The upper level performs the optimization of the task assignments. The search procedure begins with an initial task assignment set. A 2-optimal completion order [14, 16] is constructed for this task assignment set and the cost of the resulting solution is evaluated. The task assignment set is altered systematically and a new 2-optimal completion order is constructed for each new set of assignments. This search proceeds, accepting new solutions that improve upon the best cost. The description of a solution to this problem contains all of the information concerning which agents are assigned to do which tasks and the order in which these tasks are to be completed. The vehicle kinematics determine precisely when the tasks will be completed. The nonlinearity in the objective function is due to the fact that the mission time is a function of only one agent's route. This route cannot be decided *a priori* as one must know the solution to know which route is the longest one. Hence, the objective function is not a linear combination of the variables.

Algorithm 1 describes the way in which a given task assignment set and a corresponding completion order are used to determine the completion times of the tasks being serviced by the agents. This is followed by an evaluation of the algorithm's complexity. Algorithm 1 begins with a solution and loops through the corresponding completion order. The algorithm first computes the maximum arrival time of any agent performing the task T_{u_j}. It then checks the completion time of each of T_{u_j}'s precedents, p_{u_j}. Note that in line 4 of Algorithm 1, the occurrence time of a task can be equal to that of its last occurring precedent task. The constraints of (8) can be enforced strictly by including an arbitrary, fixed period of time between a task and its last occurring precedent task. The start time of this task t_{u_j} is then set to the maximum of these. Each agent performing T_{u_j} is assumed to arrive at T_{u_j} at t_{u_j}.

Data: $X = \{\mathscr{T}a, \mathscr{C}o\}$
1 **for** $j = 1$ **to** N_t **do**
2 $\quad t_{a_{\max}} = \max_i \{t_{a_{iu_j}}\}, \forall a_i \in \mathscr{A}_{a_{u_j}}$
3 $\quad t_{p_{\max}} = \max_k \{t_k\}, \forall T_k \in p_{u_j}$
4 $\quad t_{u_j} = \max\{t_{a_{\max}}, t_{p_{\max}}\}$
5 $\quad t_{a_{u_j}} = t_{u_j}, \forall a_i \in \mathscr{A}_{a_{u_j}}$
6 **end**
Result: $\{t_{u_1}, \ldots, t_{u_{N_t}}\}$

Algorithm 1: Timing evaluation

For each of the N_t tasks, the evaluations are as follows. Arrival time computation is $\mathscr{O}(N_a)$ in the worst case for checking every $a_i \in \mathscr{A}_{a_j} = \mathscr{A}$. The worst-case complexity for precedent occurrence time computation is $\mathscr{O}(N_t - 1) = \mathscr{O}(N_t)$ for checking all tasks in $\mathscr{C}o$ for each task. Setting the actual start times is performed in $\mathscr{O}(N_t)$ and $\mathscr{O}(N_a)$ time, respectively. The worst-case complexity for the total evaluation is then $\mathscr{O}(N_a N_t^2)$. The cost of any solution given in the form $X = \{\mathscr{T}a, \mathscr{C}o\}$ is $\max_j \{t_j\}$, $j = 1, \ldots, N_t$. This is evaluated in $\mathscr{O}(N_t)$ time. The cost of a solution is then evaluated in

DOI: 10.1002/rnc

$\mathcal{O}(N_a N_t^3)$ time. This is easily reduced to $\mathcal{O}(N_a N_t^2)$ time if $\max_j\{t_j\}$ is evaluated during the timing evaluation. Thus, cost evaluation for any solution to the optimization problem requires $\mathcal{O}(N_a N_t^2)$ time.

The following describes how the solution representation is helpful in guaranteeing that every solution considered obeys the assignment and order constraints imposed on the problem. These constraints are described in (8) and (10). These types of constraints appear naturally in cooperative control problems considering heterogeneous agents and complicated missions that contain order-dependent outcomes [21]. The restrictions on which agents are permitted to be assigned to each task are captured in (10). The heuristic method will only choose assignments for tasks that are from this set. This ensures that all assignments meet the constraint in (10).

The next step in guaranteeing constraint satisfaction is to use the completion order set $\mathscr{C}o$ to describe the portion of the solution that represents the order in which tasks are performed. These partially ordered sets of tasks are constructed to guarantee order constraint satisfaction. The sets are then refined using an operation that preserves constraint satisfaction. Consider m-order constraints of the form,

$$C_{p_l} = (T_j, T_k) \quad \forall l = 1, \ldots, m. \tag{11}$$

These order constraints (11) indicate that task T_j must be performed before task T_k. This input must be restructured to reflect the transitive relationships between tasks. For example, if task T_i must be performed before task T_j and T_j before task T_k, then T_i must also be performed before T_k. In this case, T_k must also be performed after T_i. The pairwise order constraints of (11) are restructured into the sets p_j and d_j of (3) and (4) for each task $T_j \in \mathscr{T}$.

The feasibility of the assignments is satisfied if at least one agent is selected to perform each task and all agents selected to perform each task $\mathscr{T}_j$ belong to $\mathscr{A}_{f_j}$. This is possible provided that the following is true:

$$\mathscr{A}_{f_j} \neq \emptyset \quad \forall T_j \in \mathscr{T}, \tag{12}$$

$$\mathscr{A}_{a_j} \neq \emptyset \quad \forall T_j \in \mathscr{T}. \tag{13}$$

Precedence constraints can be input in the form of (11). These constraints must be linked together to form the sets p_j and d_j for each task. The feasibility of the resulting constraints is checked by the following:

$$p_j \cap d_j = \emptyset \quad \forall T_j \in \mathscr{T}. \tag{14}$$

The check in (14) says that no task T_j may have the restriction that there is a task T_k that must come before and after the task T_j. For example, the constraints $C_{p_1} = (T_i, T_j)$, $C_{p_2} = (T_j, T_k)$, and $C_{p_3} = (T_k, T_i)$ result in an infeasible instance of the problem.

The completion order is a partially ordered set. Order relations are defined between constrained tasks and tasks that belong to the same agent's route. Completion orders are constructed using a greedy method. Algorithm 2 details this process. The sets *available* and *notAvailable* refer to the sets of tasks that have had all of their precedent tasks selected and those that have not, respectively. The cardinality of a set is denoted by $|\cdot|$. Algorithm 2 is initialized with an empty completion order and all tasks are initially assumed unavailable. Each unavailable task is tested for availability by checking the completion order for its precedent tasks. If it has none or if the precedent tasks are included, it becomes available. Tasks are chosen from the set $available$ according to which task will increase the final time the least. The procedure for determining available tasks is performed in $\mathcal{O}(N_t^2)$ time due to having to search $\mathscr{C}o$ for each task in *notAvailable*. The cost associated with adding each task $T_k \in$ available to the completion order is computed. The task that incurs the minimum increase in cost is added to the completion order. Each cost function evaluation is done in $\mathcal{O}(N_a N_t^2)$ time. The entire evaluation takes $\mathcal{O}(N_a N_t^4)$ time. After the completion order is constructed for the set of task assignments, 2-opt refinement is used on the route.

The 2-opt exchange heuristic is a specific case of the variable k-opt heuristic at the heart of the Lin–Kernighan heuristic [16]. The 2-opt heuristic achieves in the neighborhood of 5% excess over

DOI: 10.1002/rnc

Data: $\mathcal{T}a$
$\mathcal{C}o = \emptyset$
$notAvailable = \mathcal{T}$
$available = \emptyset$
while $|notAvailable| > 0$ **do**
  for $k = 1$ **to** $|notAvailable|$ **do**
    if $u_k \in \mathcal{C}o, \forall u_k \in p_{u_k}$ **then**
      $available = available \bigcup \{u_k\}$
      $notAvailable = notAvailable \setminus \{u_k\}$
    end
  end
  $T_{min} = argmin_{T_k} \ cost(X), T_k \in available$
  $\mathcal{C}o = \{\mathcal{C}o, T_{min}\}$
  $notAvailable = notAvailable \setminus T_{min}$
  $available = available \setminus T_{min}$
end
Result: $X = \{\mathcal{T}a, \mathcal{C}o\}$

Algorithm 2: Greedy completion order construction

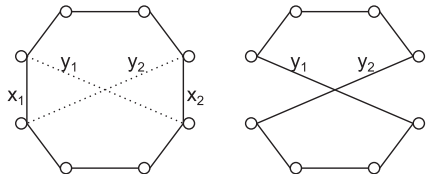

Figure 1. Standard 2-opt exchange on tours.

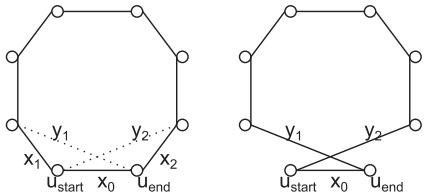

Figure 2. 2-opt exchange on vehicle routes.

the Held and Karp lower bound for single TSP instances using random Euclidean instances [14]. The basic 2-opt move cuts an agent route at two segments. Each segment is the link between two tasks or cities in a route. The problem is to find two of these segments (x_1 and x_2) that can be replaced with two different segments (y_1 and y_2) to reduce the cost of the route. This can be generalized to more than two cuts [16]. The version of the 2-opt exchange used here operates on the routes of the individual agents. The standard 2-opt exchange does not necessarily result in routes that obey order constraints. This feasibility is ensured by rejecting those routes that do not obey order constraints. A basic 2-opt move for a single closed tour (route with start position the same as the end position) is shown in Figure 1. The vehicle routes however do not necessarily begin and end at the same location. To accommodate this, the starting location and end location are connected by a link x_0 that is never considered as a candidate for the 2-opt exchange. This is shown in Figure 2. Note that for the basic 2-opt exchange, x_1 and x_2 each correspond to a pair of tasks. For any pair of cuts, x_1 and x_2, if the cuts are adjacent, a route remains unchanged. All other pairs of x_1 and x_2 are tested for improvement.

DOI: 10.1002/rnc

The process of completion order improvement by 2-opt exchange proceeds as follows. Consider the completion order,

$$\mathscr{C}o = \{T_1, T_2, T_3, T_4\}. \tag{15}$$

Let the route of agent a_1 be the totally ordered set, $\{u_{\text{start}}, T_2, T_4, u_{\text{end}}\}$ and the single agent completion order be,

$$\mathscr{C}o_1 = \{T_2, T_4\}. \tag{16}$$

The only 2-opt exchange for this route results in,

$$\mathscr{C}o_1 = \{T_4, T_2\}. \tag{17}$$

The resulting agent route is then reinserted into the original completion order to obtain,

$$\mathscr{C}o = \{T_1, T_4, T_3, T_2\}. \tag{18}$$

The resulting completion order is then tested for adherence to the precedence constraints. Resulting completion orders that violate precedence constraints are rejected and the process proceeds checking the remaining possible 2-opt exchanges until a feasible improvement is found or the possible 2-opt exchanges are exhausted. This process is performed for each vehicle's route; when an improvement for one vehicle's route is found, the improvement is kept and the process is repeated for the next vehicle's route. This process terminates when no further 2-opt improvements are possible for any route. The search through possible 2-opt exchanges requires $\mathcal{O}(N_t^2)$ time due to the worst case of searching each possible 2-opt exchange. The evaluation used to check whether a new completion order violates precedence constraints requires $\mathcal{O}(N_t^3)$ computations. The timing evaluation of Algorithm 1 requires $\mathcal{O}(N_a N_t^4)$ computations. It can now be seen that if a particular problem instance is highly constrained, the 2-optimization of completion orders will be dominated by $\mathcal{O}(N_a N_t^5)$ computations instead of $\mathcal{O}(N_a N_t^6)$ computations. This is because the violating solutions are rejected before the timings are evaluated. For highly constrained problem instances, the algorithm spends most of its time rejecting solutions instead of checking costs. This is desirable because in practice, the algorithm runs faster when more constraints are introduced.

Tabu search is a metaheuristic used for combinatorial optimization [15]. The Tabu search can be viewed as searching a graph in the neighborhood of a given initial solution. The neighborhood of a task assignment set is as follows:

$$\mathscr{N}_{\mathscr{T}a} = \{\mathscr{T}a' \mid \exists! \mathscr{A}a'_j \in \mathscr{T}a' \text{ s.t. } \mathscr{A}a'_j = \mathscr{A}a_j \Rightarrow \mathscr{T}a' = \mathscr{T}a\}. \tag{19}$$

The Tabu search relies on a move to be defined that is used to perturb one solution to another. The move used here is a perturbation in the assignment for each task. The feasible assignments $\mathscr{T}a_{f_j}$, for each task are enumerated as follows for the case of $\mathscr{A}_{f_j} = \mathscr{A}$, for all T_j:

$$\mathscr{T}a_{f_j} = \{\{a_1\}, \{a_2\}, \{a_1, a_2\}, \{a_3\}, \{a_1, a_3\},$$
$$\{a_2, a_3\}, \{a_1, a_2, a_3\}, \ldots, \{a_1, \ldots, a_{N_a}\}\}. \tag{20}$$

The set $\mathscr{T}a_{f_j}$ is the power set of $\mathscr{A}_{f_j}$ where the empty set is not considered. Each set in this family represents one or more agents being assigned to a task. The feasible set of assignments $\mathscr{T}a_{f_j}$, for each task T_j, is

$$\mathscr{T}a_{f_j} = \mathscr{P}(\mathscr{A}_{f_j}) \setminus \{\emptyset\}. \tag{21}$$

Let c_j be a bound placed on the number of agents allowed to perform task T_j. If $c_j = N_a$, the cardinality of each set of possible assignments to each task, $|\mathscr{T}a_{f_j}|$, grows exponentially with the number of agents. With c_j equal to a constant, $|\mathscr{T}a_{f_j}|$ is bounded by $\mathcal{O}(N_a^{c_j})$. The time to search the elements of $\mathscr{T}a_{f_j}$ is $\mathcal{O}(N_a^{c_j})$.

The feasible set of assignments is generated only once at the start of the optimization. The Tabu search algorithm used in this work is summarized as follows. The termination criterion for

DOI: 10.1002/rnc

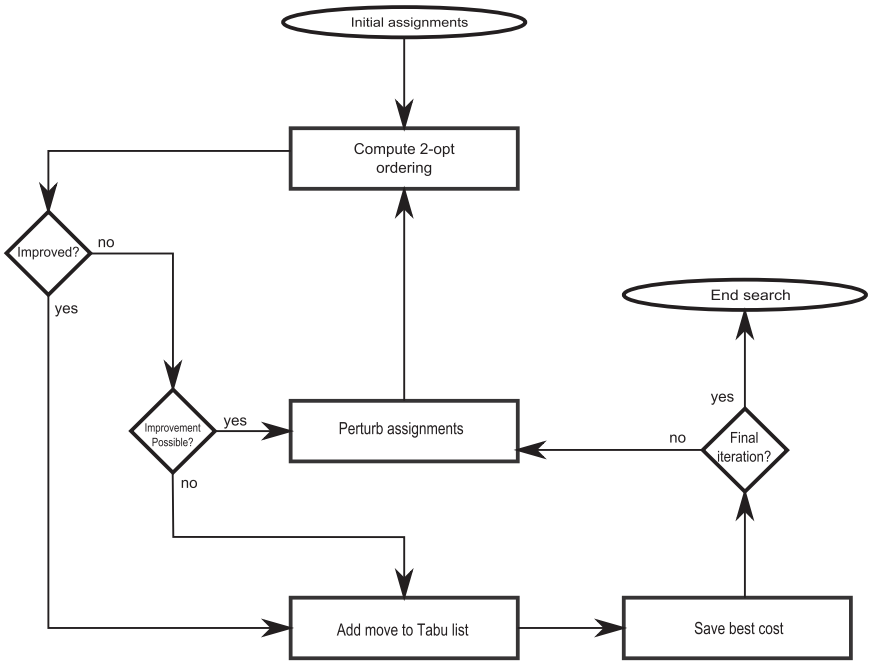

Figure 3. Overview of heuristic search.

the algorithm is reached when it has run a fixed number of iterations, $nIterations$. This number is based on a judgement of convergence by the user. The value used here was $nIterations = 100$. The Tabu search algorithm begins with an initial feasible solution X_o. The neighboring solutions X_j, $j = 1, \ldots, |\mathcal{N}_{\mathcal{T}a}|$, of the current solution X correspond to task assignment sets in the neighborhood $\mathcal{N}_{\mathcal{T}_a}$ of the task assignment set for the current solution X. The search then checks the neighboring solutions by searching the neighboring task assignment sets, generating the corresponding 2-optimal completion orders for them and checking the resulting solution X_j for a cost improvement. The search through the neighborhood of a solution stores the best solution in that neighborhood as X'. The function, *cost* evaluates the mission time for a given solution. If $\text{cost}(X_j) < \text{cost}(X_{\text{best}})$ the best solution known is replaced with X_j. In the event that the search through the neighborhood of X does not provide improvement, a local minimum has been reached and the current solution is updated to the best neighboring solution found X'. A Tabu list is kept, to prevent becoming stuck in a local minimum. When a new move is chosen, its inverse is added to the Tabu list. That is, a move that undoes that perturbation is not allowed while that move is on the Tabu list. A chart of this algorithm is presented in Figure 3.

The overall complexity at each step of the algorithm can be seen by combining our analysis of the completion order construction and refinement with the Tabu search procedure complexity. For each step of the Tabu search, the worst-case number of perturbations in the assignments at the current solution will be $\mathcal{O}(N_a^{c_j} N_t)$ because this procedure could be performed for every $T_j \in \mathcal{T}$. An upper bound on the complexity at each step is the Tabu search step complexity combined with the complexity incurred by generating the refined completion orders at each step. This gives a step complexity of $\mathcal{O}(N_a^{c_j} N_t^7)$. For the following examples, $c_j = 1$, resulting in the algorithm scaling as $\mathcal{O}(N_a N_t^7)$. With $|\mathcal{A}| \leqslant |\mathcal{T}|$, the step complexity is bounded by $\mathcal{O}(N_t^8)$.

In some cases, a Nearest Assignment heuristic is used to initialize the Tabu Search. This heuristic is simple and in many cases can provide an increase in solution quality. The Nearest Assignment heuristic initializes the task assignments as the lowest cost for each task as defined with respect to the initial configuration of the agents. This can be described as follows:

$$\mathcal{T}a_{nn_j} = \min_{\mathcal{T}a_{f_j}} \{t_{a_j}\} \tag{22}$$

DOI: 10.1002/rnc

This heuristic physically manifests as choosing the agent that is capable of reaching the task the quickest. In the special case that all agents can travel at the same maximum speed, the heuristic has the effect of partitioning the tasks geographically. This clustering effect typically gives the assignment search a better initial starting point. The search then proceeds to satisfy all ordering constraints and reassigns tasks to improve cost.

3. TREE SEARCH

A branch and bound method is used to search a tree for a solution with optimal cost. For a certain number of agents a_i where $i = \{1, 2, \ldots, N_a\}$ performing tasks on a set of targets T_j, where $j = \{1, 2, \ldots, N_t\}$, the tree is constructed by generating nodes that represent the assignment of an agent $a_i \in \mathscr{A}$ to a target $T_j \in \mathscr{T}$ at a specific time. The child nodes are found by enumerating all of the possible assignments that can be made, based on the remaining tasks and requirements of the mission. Nodes are constructed until all combinations of agents and tasks have been taken into account. Note that a branch of the tree, from a root node to a leaf node, represents a feasible set of tasks for an agent. For a scenario of three agents completing four tasks, Figure 4 shows a part of the subtree related to agent a_1. In order to represent all the possible assignments for this scenario, there should be another two subtrees related to a_2 and a_3. The top node of each of these subtrees is connected to the root node of the entire tree. Each node represents one assignment of an agent to a task where the notation S_{ij} denotes agent a_i completing task T_j. After generating the tree, an exhaustive search starts scanning the tree branches to reduce them according to which satisfy the specific mission constraints. Applying the mission constraints to the tree will reduce the possible assignments, if the agents cannot find trajectories that maintain the task precedence [13].

The Branch and Bound search algorithm is initiated by a Best First Search [28] algorithm that provides an immediate feasible assignment. It starts with the root node of the tree and the estimated cost of each child is calculated by using the distance between the assigned vehicle and its designated target. The child node with the smallest estimate is selected and the cost of the trajectory to perform the assignment is evaluated.

4. SIMULATION RESULTS

Several examples are chosen that can be treated by both the heuristic method and the optimal tree search. The computational limits for this implementation of the Branch and Bound search are reached at a problem size of about eight tasks and three vehicles. The vehicle initial positions lie on a 10×10 Euclidean grid and the vehicle maximum velocities are equal to 1 with consistent units. Tasks are to visit waypoints that lie on the 10×10 Euclidean grid. Figure 5 shows an example problem instance.

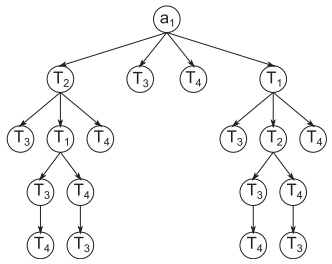

Figure 4. Tree structure.

DOI: 10.1002/rnc

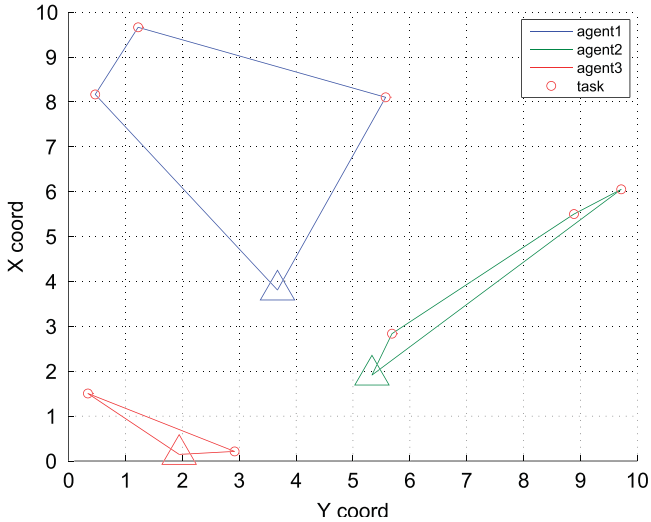

Figure 5. Example problem instance for method comparison.

Table I. Optimality gap.

Instance	Mean	Min	Max
Unconstrained (%)	23	6	39
Constrained (%)	22	9	31

The results of mean, minimum, and maximum optimality gap for 20 randomly generated problem instances are presented are Table I. The optimality gap is defined as:

$$\mathrm{OG} = \frac{\mathrm{cost}_{\mathrm{heuristic}} - \mathrm{cost}_{\mathrm{opt}}}{\mathrm{cost}_{\mathrm{opt}}}. \tag{23}$$

The solutions to the 10 unconstrained cases show a mean optimality gap of 23% and the constrained examples show a corresponding optimality gap of 22%. For each agent, the user inputs the start and desired end position as well as the types of task that the agent is allowed to perform. For each task, the inputs are the beginning and final position for the task and the type of the task. In general, the agent may begin and end the task at different positions. These results are produced from instances of the problem that are subject to the following precedence constraints:

$$C_{p_1} = (T_1, T_2), \tag{24}$$

$$C_{p_2} = (T_2, T_3). \tag{25}$$

The following example contains simulation results for a scenario where three agents perform 30 tasks. In this example there are several precedence constraints that must be obeyed. This example demonstrates the performance of the search method on a problem similar to the multiple depot mTSP. The problem is for three agents to visit 30 tasks with the cost being the maximum time for any agent to finish its route and arrive at its final location. The final location is set to be its starting location. The order constraints are,

$$C_{p_1} = (T_1, T_2), \tag{26}$$

$$C_{p_2} = (T_2, T_3), \tag{27}$$

$$C_{p_3} = (T_4, T_5), \tag{28}$$

$$C_{p_4} = (T_5, T_6), \tag{29}$$

$$C_{p_5} = (T_7, T_8), \tag{30}$$

DOI: 10.1002/rnc

$$C_{p_6} = (T_8, T_9), \tag{31}$$
$$C_{p_7} = (T_{10}, T_{11}), \tag{32}$$
$$C_{p_8} = (T_{11}, T_{12}). \tag{33}$$

Here, solutions of the same problem instance are compared. First, the starting assignment for the optimization was chosen at random. Second, the starting assignment is chosen using the nearest assignment heuristic. Figures 6(a) and (b) show the convergence of the Tabu search for the two cases obtained after 100 iterations of the Tabu search. The dotted line is the current solution that is being perturbed throughout the optimization and the solid lower bounding line is the progress of the best solution.

Figures 7(a)–(c) present the paths of the agents as they visit each task for the case of random initial assignment. It was found that when the optimization algorithm is initialized with random initial assignments, the resulting solution tends to settle into equilibria where the vehicles loop around large portions of the plane. This seems to be due to the uniform nature of the initial assignments. These solutions tend to have more gentle turns, although this is not explicitly enforced.

The times of completion of the order constrained tasks are listed in Table II. Notice that even though several tasks are coupled through constraint relations, these tasks need not be serviced by the same agent. The algorithm is capable of enforcing the transitive order constraints without requiring those tasks to be done by the same agent. The routes resulting from the use of the nearest initial agent heuristic are presented below. These routes are more efficient with regards to moving shorter distances and to the global cost function itself. However, these routes tend to have many tight turns which are not penalized by the cost function or constraints. This can be seen by the plots of the routes in Figures 8(a)–(c).

The satisfaction of the constraints is independent of the type of initial assignment and all routes are 2-optimal with respect to the restrictions of the precedence constraints. It is worth noting that the routes of the vehicles cross each other. This occurs for the route of an individual agent and in an inter-agent sense. This is particular to the cost function used. For example, if the cost were the total length of all routes and not the mission time, this would not occur when the nearest agent heuristic is used or in optimal solutions. The precedence constraints also contribute to this phenomenon. In the presence of the precedence constraints, it can be beneficial for agents' routes to cross. Figure 9(a) shows an example with no precedence constraints. Notice that two of the agents routes cross due to the choice of cost function, but that because all individual routes are 2-optimal, no individual route crosses itself. This example also shows that the nearest assignment heuristic can be improved upon by the Tabu search. Figure 9(b) shows a slight increase in the solution quality from the initial assignment. These solutions are generated in approximately 18 s on a 2.4-GHz MacBook.

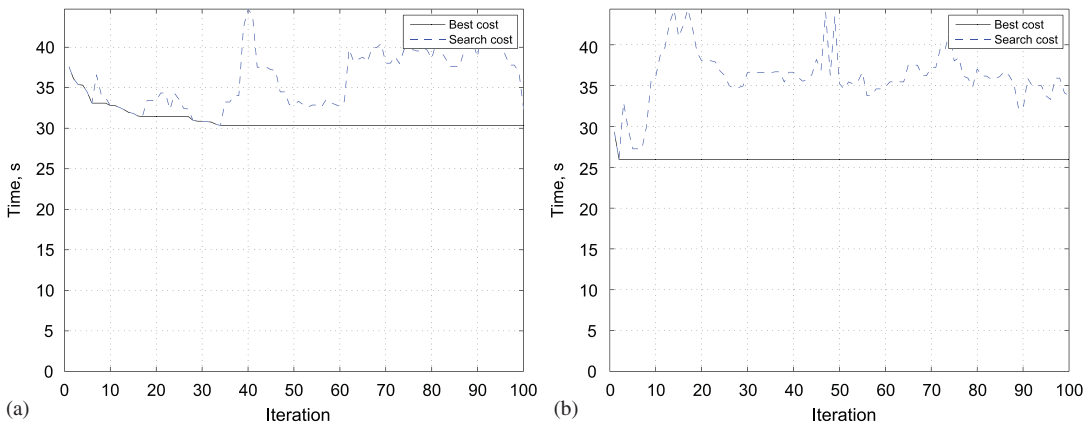

Figure 6. Tabu search cost: (a) random initial solution and (b) nearest initial assignment.

DOI: 10.1002/rnc

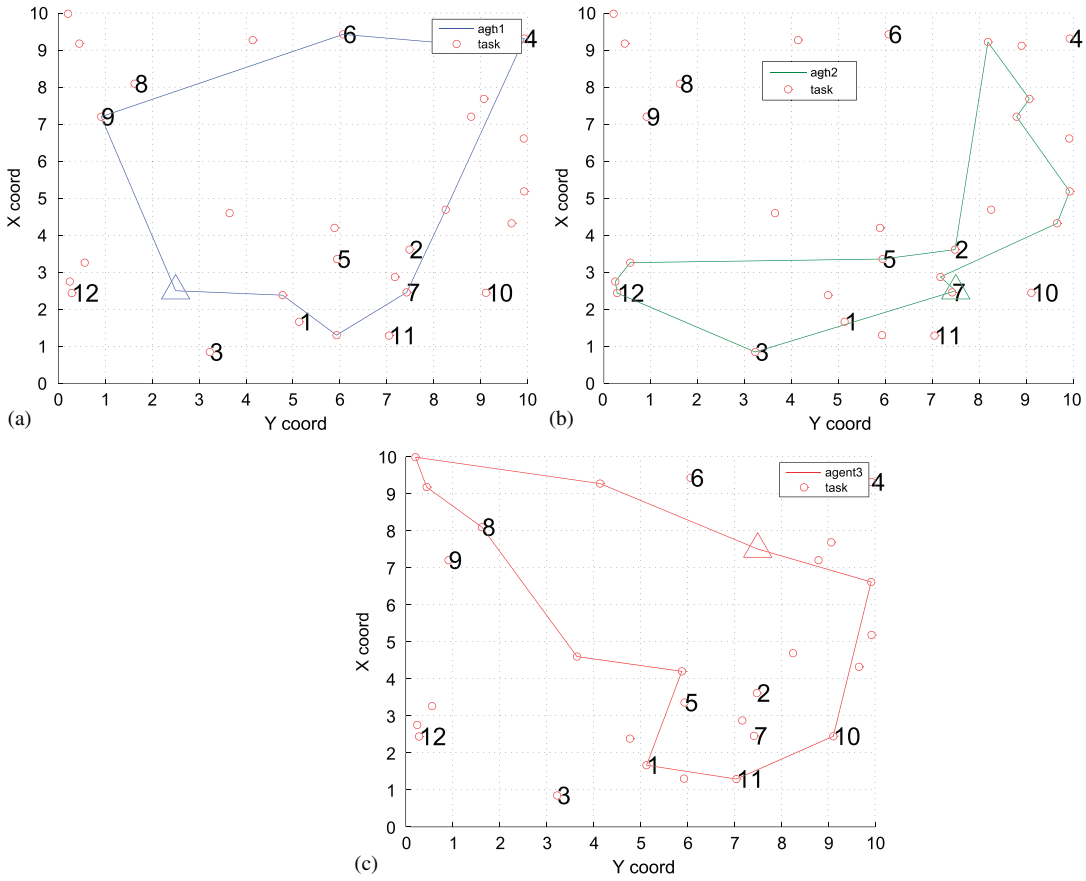

Figure 7. Agent routes and results for random initial assignment: (a) Route of agent 1; (b) Route of agent 2; and (c) Route of agent 3.

Table II. Completion times of constrained tasks.

Task	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
Completion time	11.1	14.5	25.7	13.0	16.1	16.9	5.7	20.0	22.5	6.8	9.1	22.4
Completing agent	a_3	a_2	a_2	a_1	a_2	a_1	a_1	a_3	a_1	a_3	a_3	a_2

5. CONCLUSIONS AND FUTURE WORK

The authors have developed a polynomial-time algorithm for solving a nonlinear VRP. The solution quality was compared with a Branch and Bound solution technique that provides optimal solutions for small problem instances. This analysis showed that while the combined heuristic is suboptimal, the resulting solutions average 22–23% deviation from optimal. This combined heuristic is able to plan assignments and routes quickly. The final solutions for a typical instance with 30 tasks are computed in less than 20 s on a modest computer, but the initial feasible solution is generated in about 0.004 s. This time requirement is quite acceptable for real-time operations where the mission execution time can be an hour or more. Owing to the greedy, monotonic cost improvements and to the consideration of feasible solutions only, this method is suitable for any-time operation. This is compared with a branch and bound method that is not suitable for any-time operation for larger problem instances. The result is a method that offers a user feasible solutions quickly with improvement given more time.

DOI: 10.1002/rnc

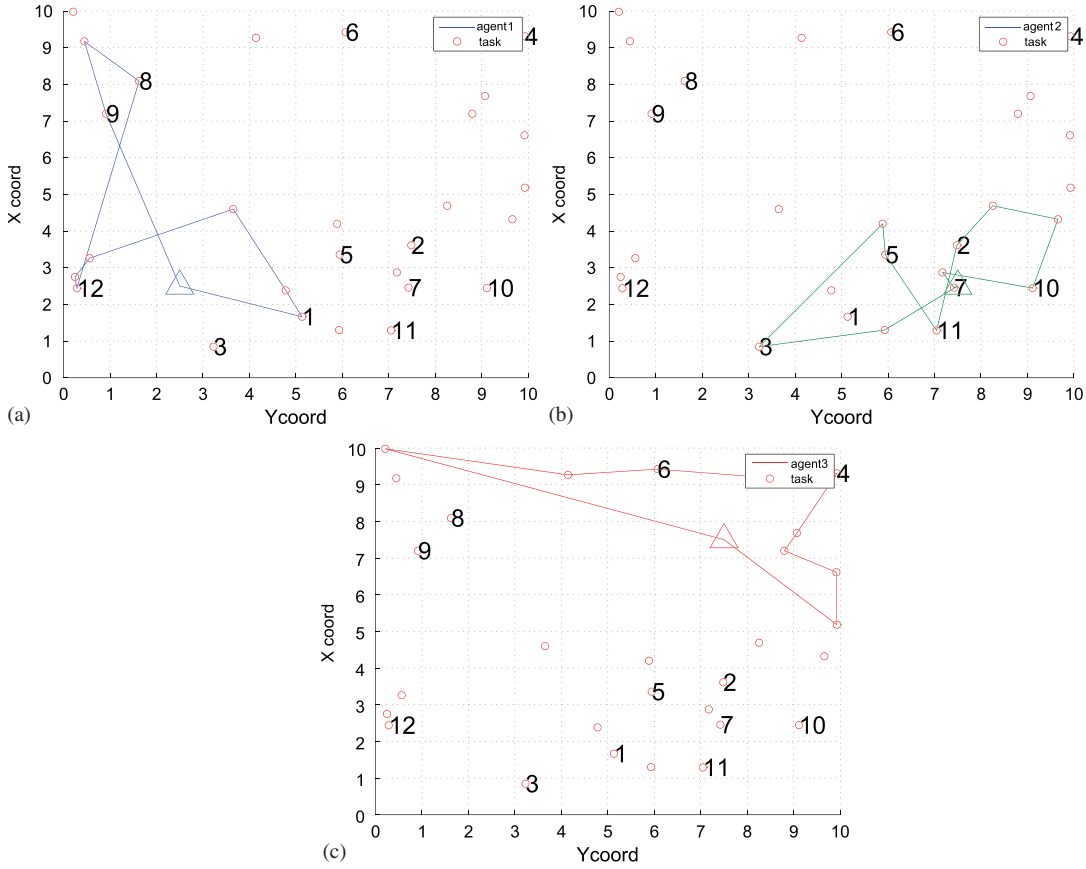

Figure 8. Agent routes and results for nearest initial assignment: (a) Route of agent 1; (b) Route of agent 2; and (c) Route of agent 3.

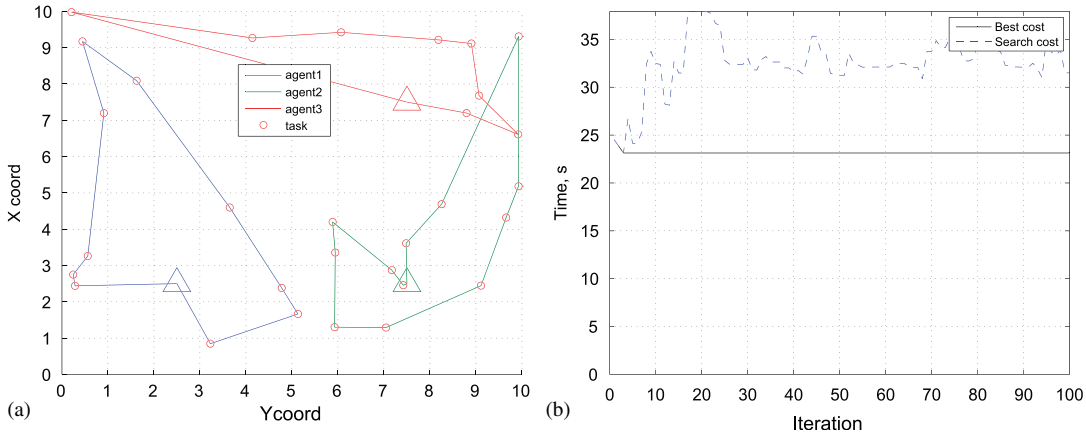

Figure 9. Example solution with no precedence constraints: (a) Routes with no precedence constraints and (b) Corresponding cost.

Some of the most restrictive assumptions of this work are associated with the vehicle kinematic model. Incorporating realistic turn-rate limitations would result in flyable trajectories that could then be assigned to unmanned aircraft or ground vehicles. Other likely extensions of this work relate to the search itself. The Tabu search can be extended to search more extensive perturbations by extending the definition of the neighborhood, $\mathcal{N}_{\mathcal{T}a}$ to include changes in the assignments of

DOI: 10.1002/rnc

more than one task. The 2-opt procedure can also be extended to include *k*-opt exchanges which would further decrease the optimality gap.

ACKNOWLEDGEMENTS

This research was partially funded by United States Air Force grant FA 8650-07-2-3744.

REFERENCES

1. Alighanbari M, Kuwata Y, How JP. Coordination and control of multiple UAVs with timing constraints and loitering. *American Control Conference*, Denver, CO, U.S.A., 2003; 5311–5316.
2. Ren W, Beard RW, Atkins EM. A survey of consensus problems in multi-agent coordination. *American Control Conference*, Portland, OR, U.S.A., 2005; 1859–1864.
3. Shima T, Rasmussen S, Gross D. Assigning micro UAVs to task tours in an urban terrain. *IEEE Transactions on Control Systems Technology* 2007; **15**:601–612.
4. Chandler P, Pachter M, Nygard K, Swaroop D. *Cooperative Control for Target Classification*. Kluwer: Dordrecht, 2001; 1–19.
5. Chandler P, Pachter M, Swaroop D, Fowler JM, Howlett JK, Rasmussen S, Schumacher C, Nygard K. Complexity in UAV cooperative control. *American Control Conference*, Anchorage, AK, 2002; 1831–1836.
6. Guo W, Nygard K. Combinatorial trading mechanism for task allocation. *International Conference on Computer Applications in Industry and Engineering*, San Francisco, CA, U.S.A., 2001.
7. Murphy R. *An Approximate Algorithm For a Weapon Target Assignment Stochastic Program*. Kluwer: Dordrecht, 1999; 1–16.
8. Nygard K, Kendall E. Dynamic network flow optimization models for air vehicle resource allocation. *American Control Conference*, Arlington, VA, U.S.A., 2001; 1853–1858.
9. Rasmussen S, Schumacher C, Chandler P. Investigation of single vs. multiple task tour assignments for UAV cooperative control. *AIAA Guidance*, *Navigation*, *and Control Conference*, Monterey, CA, U.S.A., 2002.
10. Rasmussen S. A multiple UAV simulation for researchers. *AIAA Modeling and Simulation Technologies Conference*, Austin, TX, U.S.A., 2003.
11. Schumacher C, Chandler P, Rasmussen S. Task allocation for wide area search munitions via network flow optimization. *AIAA Guidance*, *Navigation*, *and Control Conference*, Montreal, Canada, 2001.
12. Jackson J, Girard A, Rasmussen S, Schumacher C. A combined tabu search and 2-opt heuristic for multiple vehicle routing. *American Control Conference*, Baltimore, MD, 2010; 3842–3847.
13. Faied M, Mostafa A, Girard A. Dynamic optimal control of multiple depot vehicle routing problem with metric temporal logic. *American Control Conference*, St. Louis, MO, 2009; 3268–3273.
14. Aarts E, Lenstra J. *Local Search in Combinatorial Optimization*. Wiley: New York, 1997; 215–310.
15. Glover F. Tabu search—part I. *Journal of the Operations Research Society of America* 1989; **1**:190–206.
16. Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. *Technical Paper*, Bell Telephone Laboratories, Inc., Murray Hill, NJ, October 1971.
17. Helsgaun K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *Technical Report*, Roskilde University, 2000.
18. Kara I, Bektas T. Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research* 2006; **174**:1450–1458.
19. Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega The International Journal of Management Science* 2005; 209–219.
20. Ascheuer N, Junger M, Reinelt G. A branch and cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Computational Optimization and Applications* 2000; **17**:61–84.
21. Shima T, Rasmussen S, Schumacher C, Ceccarelli N, Chandler P, Jacques D, Kish B, Pachter M. *UAV Cooperative Decision and Control Challenges and Practical Approaches*. SIAM: Philadelphia, PA, 2009.
22. Faied M, Assanein I, Girard A. UAVs dynamic mission management in adversarial environments. *International Journal of Aerospace Engineering* 2009; **2009**:1–10. Article ID.107214. DOI: 10.1155/2009/107214.
23. Kirkpatrick S, Gelatt CDJ, Vecchi MP. Optimization by simulated annealing. *Science* 1983; **220**:671–680.
24. Freisleben B, Merz P. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problem. *Proceedings of IEEE International Conference*, Nagoya, Japan, 1996; 616–621.
25. Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. *Management Science* 1994; **40**:1276–1290.
26. Moccellin JV, Nagano MS. Evaluating the performance of tabu search procedures for flow shop sequencing. *Journal of the Operational Research Society* 1998; **49**:1296–1302.
27. Laguna M, Kelly JP, Gonzalez-Velarde JL, Glover F. Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research* 1995; **82**:176–189.
28. Russell S, Norvig P. *Artificial Intelligence*. Prentice-Hall: Upper Saddle River, NJ, 2010.