2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

A Combined Tabu Search and 2-opt Heuristic for Multiple Vehicle Routing

Justin Jackson*, Anouck Girard†, Steven Rasmussen‡, Corey Schumacher§
*PhD Candidate, Aerospace Engineering
University of Michigan, Ann Arbor, Michigan 48109
Email: jpjack@umich.edu
†Assistant Professor, Aerospace Engineering
University of Michigan, Ann Arbor, Michigan 48109
‡Miami Valley Aerospace LLC,
Control Science Center of Excellence
AFRL Wright-Patterson AFB, Dayton, OH 45433
§Control Science Center of Excellence
AFRL Wright-Patterson AFB, Dayton, OH 45433

Abstract— **In this paper, the authors consider the problem of assigning multiple agents to perform several tasks under restrictions limiting which agents are permitted to perform each task and restrictions dictating the order of completion of specific tasks. A unique solution description is introduced that allows for the direct satisfaction of these constraints. A two phase optimization procedure is used. A Tabu search method is used to optimize the task assignments and an exchange heuristic is used to refine the completion order of the tasks in the presence of precedence constraints.**

I. INTRODUCTION

This paper presents a combined Tabu and 2-opt heuristic for solving an assignment problem where multiple agents are assigned to perform several tasks in the presence of assignment constraints and precedence constraints. The problem considered in this paper is a variant of the vehicle routing problem (VRP) where multiple vehicles are to visit several customers. The goal of the optimization is to minimize the time of arrival of the last agent to its specified final location after all of the tasks have been performed. Solutions to the problem are described in terms of a set of agents assigned to each task and a partially ordered set dictating the completion order of the tasks. The Tabu search operates on the task assignments, where a cost is assigned based on the best completion order found for the given task assignments using a 2-opt exchange heuristic to optimize the routes of the agents. After the cost is computed for the task assignments the task assignments are perturbed by changing the assignment for a single task and reevaluating the cost. If the perturbation has lower cost, it is successful and is accepted. This method searches locally, but accepts the best non-improving move if no improving moves are available. A Tabu list is incorporated to prevent a return to a previously visited local minimum. This method guarantees adherence to many precedence constraints and assignment constraints, and provides substantially refined solutions when starting from a random initial guess in modest computation time.

A. Original contribution

The authors give a solution representation that easily satisfies assignment and order constraint satisfaction. The solution method presented searches only feasible solutions. This combined heuristic method is an original combination of two types of common combinatorial search methods. This procedure is able to effectively solve the optimization problem in the presence of nonlinearities introduced by a nonlinear cost function and by the precedence constraints. In addition, the timing associated with all precedence constraints is obeyed even when constrained tasks are performed by different agents.

B. Motivation

Cooperative control problems are often viewed in the framework of task assignment or vehicle routing problems [14]. Many applications require the control of multiple agents for the completion of several tasks, e.g. manufacturing tasks, the pick up and delivery of packages or people, and the time critical scheduling of cooperative military missions. Often, these tasks have a precedence order associated with the tasks such as the constraint that a package be picked up before it is delivered and the necessity to classify a target of interest before it is attacked. This paper provides a solution representation that directly satisfies assignment constraints restricting which agent can perform certain tasks. For example, a drilling machine would not be used to roll steel, a small delivery van could not be used to pick up and deliver a car, and an aircraft sensor platform would not be used to bomb a target. A combined heuristic is presented that searches for solutions in a systematic way, guarantees convergence to a local optimum, and adheres to all constraints.

C. Literature review

The VRP literature is rich with many methods and techniques for solving routing problems. The basic traveling

978-1-4244-7427-1/10/$26.00 ©2010 AACC

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 traveled. 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 typically scales exponentially with the number of locations to be visited. This problem is often formulated using linear programming formulations. There are also heuristics available to solve the problem approximately [1]. The most effective heuristic is the Lin-Kernighan variable k-opt exchange heuristic [12] and it has a very effective implementation by Helsgaun [8].

One important variation of this problem is the multiple traveling salesman problem (mTSP). This variation is concerned with multiple salesmen tasked with visiting several locations. This problem has also been addressed in the literature through linear programming formulations [9], [3] and through the use of transformations to the single TSP. This problem has been posed as the single and 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 and Junger [2] 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 treated recognizing that the relative timing of the task completion must be obeyed when separate agents perform tasks that have a precedence relation defined between them.

Other methods used for UAV routing use capacitated transshipment assignment formulations [14]. 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 UAV routing problems [14] and is a versatile tool, but scales exponentially with the number of tasks and agents. The branch and bound method has been used to solve complex UAV routing problems exactly [4]. Such representations can involve complex cost functions and environmental factors such as wind and adversarial action.

Metaheuristics such as Tabu Search [7], Simulated Annealing [10] and Genetic Algorithms [5] have been used to solve VRPs. These metaheuristics are most useful when planning in the presence of many types of constraints and when evaluating solutions with unusual or complicated cost functions. The Tabu search heuristic has been used to solve vehicle routing [6], flow shop scheduling [13], and generalized assignment problems [11]. These heuristics have the advantage of not requiring a restrictive structure for the solution or the cost function, but have the disadvantage of minimal or no guarantee on the solution quality.

D. Overview

This paper is organized as follows, section II presents the problem addressed by this work along with the major assumptions. Section III details how solutions to the problem are represented in a way convenient for constraint satisfaction. Section IV discusses how the constraints on vehicle assignment and task ordering are specified. Section V details the construction of task completion order and the 2-opt method used to refine the completion order. Section VI shows how the cost function is evaluated from a task assignment and completion order. Section VII details the Tabu search method used to optimize the task assignment and the task completion order. Sections VIII and IX present two examples used to demonstrate this method, conclusions drawn from the results and a discussion of future work.

II. Problem definition

A. Problem definition

Consider an assignment problem where N_a agents are to be assigned to complete N_t tasks. The available agents are,

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

and

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

is the set of all 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 and later in time respectively than a given task. The precedent and dependent sets are

$$p_j = \{T_k \in \mathcal{T} \mid t_k \leq t_j\}, \tag{3}$$

$$d_j = \{T_k \in \mathcal{T} \mid t_k \geq t_j\}, \tag{4}$$

and capture all transitivity relationships. The occurrence time of a task T_j, is t_j. Let the sets $\mathcal{A}_{f_j}$ and $\mathcal{A}_{a_j}$ be the set of agents allowed to perform task T_j and the set of agents that are assigned to perform task T_j respectively. The arrival time of agent a_i at its final location is $t_{a_{if}}$. The optimization problem is,

$$\min_{\mathcal{T}a, \mathcal{C}o} \max(t_{a_{if}}) \tag{5}$$

s.t.

$$t_k \geq t_j \quad \forall T_j \in p_k, \quad j = 1, \ldots, N_t, \tag{6}$$

$$t_k \leq t_j \quad \forall T_j \in d_k, \quad k = 1, \ldots, N_t, \tag{7}$$

$$t_k \geq t_{a_{ik}} \quad \forall a_i \in \mathcal{A}_{a_k}, i = 1, \ldots, N_a, \tag{8}$$

$$\mathcal{A}_{a_j} \subseteq \mathcal{A}_{f_j}. \tag{9}$$

This formulation is flexible enough to allow agents to cooperate to complete individual tasks if this helps minimize the cost. The sets $\mathcal{T}a$ and $\mathcal{C}o$ are the set of agent assignments to all tasks and the partially ordered set dictating the constrained order of completion of the tasks. The first two constraints enforce that each task must occur later than all of its precedents and before all of its dependents. The third constraint enforces that each task occur at least as late as the final arrival time of the latest agent performing the task. The last constraint restricts which agents perform each task.

B. Assumptions

Certain properties of the agents and the tasks are assumed throughout this work. For simplicity, the agents are assumed to be unconstrained by fuel limits or any other 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. Agents move with a maximum velocity in the Euclidian plane, are unconstrained by turning rate limits, and can meet waiting requirements.

III. Representation of the solution

A solution to the task assignment problem in section II-A is given in the form of the sets of agents assigned to complete each task and a partially ordered set telling the order in which the tasks should be completed. These two objects combined with the known maximum velocity capabilities of the agents contain all information to specify the time of occurrence and which agent performs each task.

$$\mathcal{T}a = \{\mathcal{A}_{a_1} \dots \mathcal{A}_{a_{N_t}}\}, \quad (10)$$

$$\mathcal{C}o = \{u_1 \dots u_{N_t}\}. \quad (11)$$

where the u_i's represent the partially ordered list of tasks. This representation is designed to satisfy specific, high level constraints on the possible solutions. The first type of constraint is a restriction on which tasks each agent is allowed to perform. The second is a restriction on the order in which tasks can be performed. The solutions considered are solutions that can be represented in terms of a task assignment component and a completion order component. The completion order gives the order in which tasks are to be completed. There are other types of constraints that cannot be directly addressed by this structure, specifically, endurance constraints and limits on resources. Note that timing constraints are not explicitly incorporated in the completion order. The timing constraints are enforced when evaluating the cost of a solution.

IV. Assignment and order constraints

A. Assignment constraints

The constraints considered in the construction of the solutions are restrictions on assignments and the order of task completion. The restrictions on which agents are permitted to be assigned to each task are captured in the following constraints,

$$\mathcal{A}_{a_j} \subseteq \mathcal{A}_{f_j}, \quad (12)$$

where $\mathcal{A}_{f_j}$ are given as an input to the optimization. It is required that the set of agents that are assigned to perform a task T_j, $\mathcal{A}_{a_j}$, be from the feasible set of agents for that task, $\mathcal{A}_{f_j}$.

B. Completion order constraints

There are m order constraints of the form,

$$C_{p_m} = (T_j, T_k). \quad (13)$$

Eq. 13 indicates 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 T_i must be performed before T_j and T_j before T_k, then T_i must also be performed before T_k. In this case, objective T_k must also be performed after T_i. The pairwise order constraints of Eq. 13 are restructured into the sets p_j and d_j of Eqs. 3 and 4 for each task $T_i \in \mathcal{T}$.

C. Feasibility definition

The definition of feasibility concerns the assignment and order constraints. The constraints of sections IV-A and IV-B result in a feasible problem if

$$A_{f_j} \neq \emptyset \quad \forall T_j \in \mathcal{T} \quad (14)$$

and

$$p_j \cap d_j = \emptyset \quad \forall T_j \in \mathcal{T}. \quad (15)$$

At least one agent must be available to perform each task. And 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 problem.

V. Completion order construction and refinement

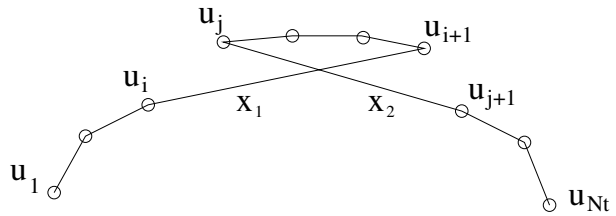
A completion order $\mathcal{C}o$, corresponding to a task assignment $\mathcal{T}a$, completely describes what tasks each agent will perform and in what order. This section presents the method used to create completion orders given a task assignment and the 2-opt exchange heuristic that is used to improve these completion orders.

A. Completion order construction for order constraint satisfaction

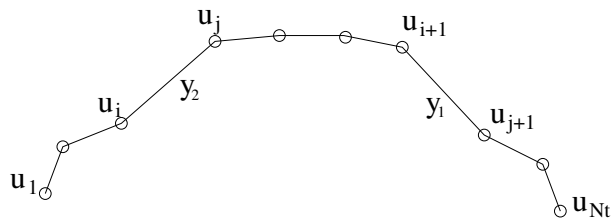
Completion orders are constructed using a greedy method. The tasks available to be placed in the ordering are determined by which tasks have previously been placed. The precedent and dependent sets of Eqs. 3 and 4 are used to determine which tasks are available. If all of a task's precedents have been selected, it is available for selection. The cost of adding each available task is computed based on which agents perform the task and on the time cost associated with the task given the history of the mission. Tasks are selected from the set of available tasks based on which task will increase the cost of the solution the least; this is the greedy selection criterion. After the completion order is constructed for the set of task assignments, 2-opt refinement is used to refine the route.

B. 2-opt refinement of agent routes

The 2-opt exchange heuristic is a specific case of the variable k-opt heuristic at the heart of the Lin-Kernighan heuristic [12]. The 2-opt heuristic achieves in the neighborhood of 5% excess over the Held and Karp lower bound for single TSP instances using random Euclidian instances [1]. The version of the 2-opt exchange used here operates on the routes of the individual agents. A basic 2-opt move is shown



(a) Route before 2-opt move



(b) Route after 2-opt move

Fig. 1. 2-opt illustration.

in figures 1(a) - 1(b). For the route of a single agent, the 2-opt move manifests as follows. Let,

$$\mathcal{C}o_1 = \{u_1, \ldots, u_i, u_{i+1}, \ldots, u_j, u_{j+1}, \ldots, u_{N_t}\}. \quad (16)$$

be the totally ordered set representing the route of an agent. Let, $x_1 = (u_i, u_{i+1})$, $x_2 = (u_{j+1}, u_j)$, $y_1 = (u_{i+1}, u_{j+1})$, $y_2 = (u_j, u_i)$. The resulting route for the agent, following the 2-opt move is

$$\mathcal{C}o_1 = \{u_1, \ldots, u_i, u_j, \ldots, u_{i+1}, u_{j+1}, \ldots, u_{N_t}\}. \quad (17)$$

Selection of the second junction, x_2 proceeds from right to left. Our version of the 2-opt exchange only makes exchanges that respect the precedents and dependents of each task. In other words, a 2-opt move is rejected if the resulting $\mathcal{C}o$ violates precedence constraints. Consider the following completion order,

$$\mathcal{C}o = \{T_1, \mathbf{T_2}, T_3, \mathbf{T_4}, \mathbf{T_5}, \mathbf{T_6}, T_7, T_8, \mathbf{T_9}, \mathbf{T_{10}}\}, \quad (18)$$

where the bold entries correspond to the following agent route,

$$\mathcal{C}o_1 = \{T_2, T_4, T_5, T_6, T_9, T_{10}\}. \quad (19)$$

Let $x_1 = (T_6, T_9)$, that is, the route will be split at the junction of T_6 and T_9. Another junction must be found to complete the 2-opt move. Let the task T_2 be a precedent of T_3, $T_2 \in p_3$. Searching right to left, the possible junctions for x_2 are $x_2 = (T_4, T_5)$ and $x_2 = (T_2, T_4)$. The junction $x_2 = (T_{10}, T_2)$ would result in a route of,

$$\mathcal{C}o_1 = \{T_6, T_5, T_4, T_2, T_9, T_{10}\}, \quad (20)$$

and a completion order of,

$$\mathcal{C}o = \{T_1, \mathbf{T_6}, T_3, \mathbf{T_5}, \mathbf{T_4}, \mathbf{T_2}, T_7, T_8, \mathbf{T_9}, \mathbf{T_{10}}\}, \quad (21)$$

which violates the precedence constraints, $T_2 \in p_3$. The possible 2-opt moves correspond to $x_1 = (T_4, T_5)$, $x_2 = (T_6, T_9)$ or $x_1 = (T_2, T_4)$, $x_2 = (T_6, T_9)$ with the following agent routes,

$$\mathcal{C}o_1 = \{T_2, T_4, T_6, T_5, T_9, T_{10}\}, \quad (22)$$

$$\mathcal{C}o_1 = \{T_2, T_6, T_5, T_4, T_9, T_{10}\}, \quad (23)$$

$$(24)$$

resulting in the completion orders,

$$\mathcal{C}o = \{T_1, \mathbf{T_2}, T_3, \mathbf{T_4}, \mathbf{T_6}, \mathbf{T_5}, T_7, T_8, \mathbf{T_9}, \mathbf{T_{10}}\}, \quad (25)$$

$$\mathcal{C}o = \{T_1, \mathbf{T_2}, T_3, \mathbf{T_6}, \mathbf{T_5}, \mathbf{T_4}, T_7, T_8, \mathbf{T_9}, \mathbf{T_{10}}\}. \quad (26)$$

VI. Cost function evaluation

Recall that the only notion of time that the task orderings retain is the precedent and dependent relationships of section II. Completion times of the tasks are required in order to find the maximum completion time of all the tasks within a solution and assign a cost to that solution as well as to insure synchronization of task completion when multiple agents are assigned to a single task. Recall that a solution tells which agents perform a task and the order in which those tasks are to be completed. Note that if all tasks in the completion order, $\mathcal{C}o$, are done sequentially, undue restrictions are created, as the only restrictions that need be respected are those of Eqs. 6 and 7. Recall that tasks have zero duration and only travel time is considered. Also consider that the agents travel the Euclidian distance between tasks and that there is no cost for turning. The following procedure is used to determine the time of completion of each task.

Data: $\mathcal{T}a, \mathcal{C}o$
for $i = 1$ **to** N_t **do**
  Compute arrival time of latest agent to arrive at u_i.
  Compute latest occurrence time of precedent tasks, $\max(t_j), T_j \in p_{u_i}$.
  Set maximum of the two as occurrence time of u_i.
  Set last arrival time of all agents $a_j \in \mathcal{A}_{a_{u_i}}$, to t_{u_i}.
end
Result: $cost = \max(t_{a_{jf}})$

Algorithm 1: Solution cost evaluation

VII. Tabu search formulation

Tabu search is a metaheuristic used for combinatorial optimization [7]. The Tabu search can be viewed as searching a graph in the neighborhood of a given initial solution. Here, the neighborhood is defined by all solutions that differ from the current solution by the assignment for a single task. The Tabu search begins with an initial feasible set of task assignments. This initial set of assignments is then perturbed to a new set of assignments in its neighborhood. A completion order is found and the cost evaluated. If the perturbation results in a better solution, the new best solution is saved and the optimization continues. If no better solution is found through perturbing the current set of assignments, a local minimum has been reached. In the latter case, the

best non-improving move that is not in the tabu list is used as the next move and the optimization continues. 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, any single move that undoes that perturbation is not allowed while that move is on the Tabu list. Each task assignment set, $\mathcal{T}a$ has a completion order associated with it. The completion order for a task assignment is computed using the method presented in section V and evaluated using algorithm 1. 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 possible assignments for each task are enumerated as follows.

$$\mathcal{T}a_p = \{\{a_1\},\{a_2\},\{a_1,a_2\},\{a_3\},\{a_1,a_3\}, \{a_2,a_3\},\{a_1,a_2,a_3\},\ldots,\{a_1,\ldots,a_{N_a}\}\} \quad (27)$$

The set $\mathcal{T}a_p$ is the power set of $\mathcal{A}$ 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 $\mathcal{T}a_{f_j}$, for each task T_j, is

$$\mathcal{T}a_{f_j} = \mathcal{P}(\mathcal{A}_{f_j}) - \emptyset \quad (28)$$

The feasible set of assignments are generated only once at the start of the optimization. A move is defined as changing an assignment for a single task to any other assignment for that task. At each iteration of the optimization, a change in the assignment for each task is evaluated to explore whether or not a cost decrease can be gained by a change of the assignment for any single task. The Tabu search algorithm used in this work is summarized in algorithm 2.

Data: Initial solution
Create library of feasible assignments for each task, $\mathcal{T}a_{f_i} \quad i = 1,\ldots,N_t$.
x = current solution
$x*$ = best solution.
$x* = x$
for $i = 1$ **to** $nIterations$ **do**
  while *there are untried perturbations* **do**
    $Improvement = 0$.
    for $j = 1$ **to** N_t **do**
      Perturb current solution at T_j.
      Evaluate best completion order for perturbed $\mathcal{T}a$.
      Evaluate cost of task assignment, cost(perturbed).
      if *cost(perturbed)* $<$ *cost(best)* **then**
        best solution = perturbed solution.
        current solution = perturbed solution.
        Improvement = 1.
        Exit for loop.
      end
      if *cost(perturbed)* $<$ *cost(current)* **then**
        current solution = perturbed solution.
        Improvement = 1.
        Exit for loop.
      end
    end
    if *Improvement* **then**
      Exit while loop.
    end
    Save best perturbation.
  end
  if *Improvement = 0* **then**
    current solution = best perturbed solution.
  end
end
Result: Best locally optimal solution

Algorithm 2: Outer loop Tabu search

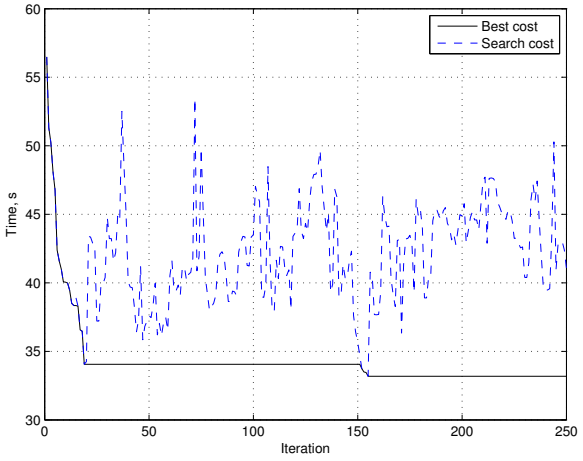


Fig. 2. Optimization cost history for cooperative search example.

VIII. Simulation Results

This section contains simulation results for a scenario where two agents perform thirty 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 with precedence constraints. The example differs in that the cost function and precedence constraints both introduce nonlinearities. The problem is for two agents to visit 30 tasks with the cost being the maximum time of either 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,

$$C_{p_1} = (T_1, T_2), \quad (29)$$
$$C_{p_2} = (T_2, T_3), \quad (30)$$
$$C_{p_3} = (T_5, T_6), \quad (31)$$
$$C_{p_4} = (T_6, T_7), \quad (32)$$

The starting solution for the optimization was chosen at random. Figure 2 shows the convergence of the cost from the initial condition to the final answer obtained after 250 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. Figure 3 presents the paths of the agents as they visit each task. The final times of each agent arriving at their respective final positions are $33.13s$ for agent 1 and $33.18s$ for agent 2. This is a local minimum in that no single assignment can be changed to decrease the solution cost. The Tabu search seeks to minimize the cost of the longest route and does so by trading tasks between the agents. The runtime of this example was $2.63s$ on a 2.4 GHz Macintosh. This low runtime is a result of the constraints. This algorithm ignores solutions that violate precedence constraints and thus does not evaluate the cost of these solutions.

The times of completion of the order constrained tasks are listed in table I. We see that even though several tasks are

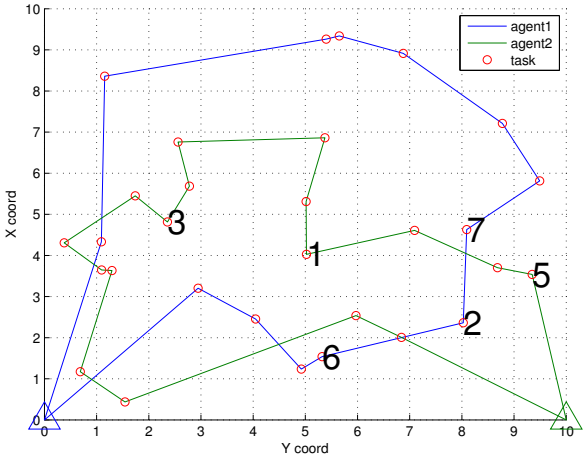


Fig. 3. Agent routes for cooperative search example.

TABLE I

COMPLETION TIMES OF CONSTRAINED TASKS.

Task	Completion time (seconds)	Completing agent
T_1	8.26	a_2
T_2	10.51	a_1
T_3	16	a_2
T_5	3.6	a_2
T_6	7.68	a_1
T_7	12.78	a_1

coupled through constraint relations, these tasks need not be done by the same agent. The algorithm is still capable of enforcing the transitive order constraints.

IX. Conclusions and Future Work

This paper addressed an assignment problem concerned with coordinating several agents in performing several tasks under precedence and assignment constraints while minimizing the time of arrival of the last agent. A solution representation was introduced that allows for the direct satisfaction of these constraints. The Tabu search method was used very naturally to optimize over the set of possible agent to task assignments and a 2-opt refinement technique that respects the precedence constraints was used to optimize the order of completion for the tasks.

Future work includes investigating cluster algorithms as a possible preprocessor to determine better initial assignments to warm-start the algorithm. This is potentially a method starting near a local minimum of relatively low cost. The authors also plan to include durations of the tasks and investigate using this algorithm for dynamic mission planning.

X. Acknowledgements

This research was supported in part by the United States Air Force grant FA 8650-07-2-3744.

References

[1] E. Aarts and J. Lenstra, *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997, pp. 215–310.

[2] N. Ascheuer, M. Junger, and G. Reinelt, "A branch and cut algorithm for the asymmetric traveling salesman problem with precedence constraints," *Computational Optimization and Applications*, vol. 17, pp. 61–84, 2000.

[3] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega The International Journal of Management Science*, pp. 209–219, January 2005.

[4] M. Faied, I. Assanein, and A. Girard, "Uavs dynamic mission management in adversarial environments," *International Journal of Aerospace Engineering*, pp. 1–10, 2009.

[5] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problem," in *Proceedings of IEEE International Conference*, Nagoya, Japan, May 1996, pp. 616–621.

[6] M. Gendreau, A. Hertz, and G. Laporte, "A tabu search heuristic for the vehicle routing problem," *Management Science*, vol. 40, pp. 1276–1290, October 1994.

[7] F. Glover, "Tabu search - part I," *Journal of the Operations Research Society of America*, vol. 1, pp. 190–206, June 1989.

[8] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," Roskilde University," Technical Report, 2000.

[9] I. Kara and T. Bektas, "Integer linear programming formulations of multiple salesman problems and its variations," *European Journal of Operational Research*, vol. 174, pp. 1450–1458, 2006.

[10] S. Kirkpatrick, J. C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.

[11] M. Laguna, J. P. Kelly, J. L. Gonzalez-Velarde, and F. Glover, "Tabu search for the multilevel generalized assignment problem," *European Journal of Operational Research*, vol. 82, pp. 176–189, 1995.

[12] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Bell Telephone Laboratories, Inc., Murray Hill, N.J., Technical Paper, October 1971.

[13] J. V. Moccellin and M. S. Nagano, "Evaluating the performance of tabu search procedures for flow shop sequencing," *Journal of the Operational Research Society*, vol. 49, pp. 1296–1302, December 1998.

[14] T. Shima, S. Rasmussen, C. Schumacher, N. Ceccarelli, P. Chandler, D. Jacques, B. Kish, and M. Pachter, *UAV Cooperative Decision and Control Challenges and Practical Approaches*. Society for Industrial and Applied Mathematics, 2009.