A new UAV assignment model based on PSO

Conference Paper · October 2008

DOI: 10.1109/SIS.2008.4668282 · Source: IEEE Xplore

4 authors, including:

Feng Pan
Beijing Institute of Technology
90 PUBLICATIONS **500** CITATIONS
SEE PROFILE

Xiaohui Hu
General Electric
24 PUBLICATIONS **3,025** CITATIONS
SEE PROFILE

Yaobin Chen
Indiana University-Purdue University Indianapolis
147 PUBLICATIONS **2,227** CITATIONS
SEE PROFILE

2008 IEEE Swarm Intelligence Symposium
St. Louis MO USA, September 21-23, 2008

A New UAV Assignment Model Based on PSO

Feng Pan [1] Xiaohui Hu [2] Russ Eberhart [2] Yaobin Chen [2]

[1] Purdue School of Engineering and Technology, Indianapolis Indiana

[2] Computelligence LLC, Indianapolis, Indiana

Email: rce@computelligencellc.com

Abstract - An unmanned aerial vehicle (UAV) assignment model requires allocating vehicles to targets to perform various tasks. It is a complex assignment problem with hard constraints, and potential dimensional explosion when the scenarios become more complicated and the size of problems increases. In this paper, a new UAV assignment model is proposed which reduces the dimension of the solution space and can be easily adapted by computational intelligence algorithms. In the proposed model a local version of particle swarm optimization (PSO) is applied to accomplish the optimization work. Numerical experimental results illustrate that it can efficiently achieve the optima and demonstrate the effectiveness of combining the model and a local version of PSO to solve complex UAV assignment problems.

I. INTRODUCTION

The innovation of advanced materials, engine and sensor technology has improved the tactical capabilities of Unmanned Air Vehicles (UAVs) and made it necessary to deal efficiently with large amounts of information. Artificial intelligence and computational intelligence methods can enable UAVs to be more intelligent and better adapt to changes in the battlefield. The roles and complexities of UAVs are growing and research approaches are evolving.

UAV assignment and task allocation problems, among the most studied topics, decompose the optimization of air-to-ground operations into several parts and study how to allocate and schedule UAVs to perform tasks so as to maximize effectiveness of the overall mission, involving goal assignment, trajectory optimization, and time or task requirements [1]. The effort consists of two main aspects: modeling, and solving task assignment problems [2].

1) Modeling the multi-task and multi-assignment problem.

Obviously, UAV assignment problems are similar and related to some typical assignment and scheduling problems. Some of these models have been adapted to the research of UAV assignment problems, including Mixed-Integer Linear Programming (MILP) [3-5], Binary Linear Programming (BLP) [6], Linear Ordering Problem (LOP) [7], Traveling Salesman Problem (TSP) [8, 9], Vehicle Routing Problem (VRP) [3, 9-11], Dynamic Network Flow Optimization (DNFO) [12, 13], and so on.

(2) Solving the multi-task and multi-assignment problem.

Approaches applied to solve UAV assignment problems can be classified into two categories: deterministic and stochastic methods. Under some assumptions, the assignment problems are simplified as a certain types of mathematical models (e.g. linear programming models), and those deterministic methods, well-developed in optimization theory and operational research, such as the Hungarian algorithm [14], dynamic programming [15], branch and bound and decision tree [16], etc., can arrive at optimal or good decisions efficiently. However, in the class of problems known as typical NP-problems, the complexity and computation costs rapidly escalate subject to the complexity increase of the assignment problem.

Stochastic methods, especially heuristic algorithms, usually don't need polynomial time to solve assignment problems. They optimize globally or sub-optimally with limited computation costs. Genetic algorithms [7, 17, 18], ant colony algorithms [19], tabu search [5, 9, 11], auction algorithms [1], particle swarm optimization [8], etc., are widely utilized to perform such optimization tasks.

In this paper, a new model of UAV assignment problems which can be easily adapted by stochastic methods is proposed and analyzed in the 2nd section. Constraints treatment is discussed in the 3rd section. Particle swarm optimization (PSO) is applied to the UAV assignment problem in the 4th section. Numerical experiments are discussed and results presented in the 5th section

II. UAV ASSIGNMENT PROBLEM FORMULATION

A. UAV Problem Definition

An instance of a UAV assignment problem consists of M UAVs which will perform K tasks at all N targets, subject to all constraints. According to various requirements, assignment schemes may be different from each other. In this paper, the first objective is to assign UAVs to minimize the total mission time cost based on an associated time cost matrix T. The second objective is to minimize the time required to calculate the mission plan.

Assumption:

(1) The number of tasks are the same as those in [3]. Three-task scenarios (classification, attack, and verification) and two-task scenarios (without classification tasks) are implemented. Four-task scenarios (with "search tasks") are studied in some papers [1] but not included in this paper.

(2) All UAVs are homogeneous. That means that all vehicles can perform all tasks on all targets, but the time consumption t_{ij} of every UAV is different

(3) The number of targets and UAVs obey the requirement in [3], which means the number of UAVs is no less than the number of targets.

(4) All targets have the same priority. In Vijay *et al.* [9], preference differences are considered. We will consider these in our future research.

Definition: The objective function can be defined as:

$$J^*(\pi_{K,M}, o_{NK}) = \min_{\substack{\pi_{K,M}\in\Pi_{K,N} \\ o_{NK}\in O_{NK}}} \{J(C_{N,M}, \Pi_{K,N}, O_{NK})\}$$
$$s.t. \qquad \pi_{K,N}, o_{NK} \in \Gamma \tag{1}$$

subject to all the constraints Γ which are listed in Table 1 and variables which are explained in Table 2.

Table 1: Constraints of UAV Problem

	Constraints [3]
1)	Mission completion requires that all tasks (two or three, depending on scenario) are performed on each target exactly one time.
2)	Not more than one UAV is assigned to perform a specific task on a specified target.
3)	A UAV, coming from the outside, can visit the target at most once; if a UAV visits a target to perform a classification, it can perform the 'attack' task on the same target.
4)	An UAV leaves a node at most once.
5)	An UAV can be assigned to attack at most one target and cannot subsequently be assigned to perform any other tasks at targets.
6)	If an UAV visits a target for the purpose of performing a 'CLASSIFICATION or 'VERIFICATION' task, it must also exit from the target.
7)	If an UAV is not assigned to visit a node, then it cannot possibly be assigned to fly out of the node.
8)	All UAVs leave the source nodes. A UAV leaves the source node even if this entails a direct assignment to the sink node.
9)	A UAV can leave a node, unless it completes the attack task at the node.
10)	A UAV can perform a task only if the preceding task at the node has been completed.

Table 2: Nomenclature

N	Number of targets
M	Number of UAVs
K	Number of tasks for each target
T_{ij}	$= \{t_{ij}\}_{(N+M)\times N}$, time cost matrix. t_{ij} is the flight time from node i to node j
C_{NM}	$= \{c_{n,m}\}_{N\times M}$, cumulative time matrix. $c_{n,m}$ stands for the cumulative time of both the n-th target and the m-th vehicle.
O_{NK}	$= \{o_{i\text{-}k}\}_{1\times N\cdot K}$, target sequence array. o_{ik} means the execution of the k-th task at the i-th target
Π_{KN}	$= \{\pi_{k,n}\}_{K\times N}$ UAV assignment matrix, $\pi_{k,m}$ is the vehicle number to perform k-th task at the n-th target
Γ	$= \{f_r(\cdot)\mid r=1,2\ldots R\}$ is the constraint set. $f_r(\cdot)$ is the r-th constraint and R is the number of constraints

B. Cumulative time matrix C_{NM}

The matrix C_{NM} (as shown in Eq.(2)) contains a cumulative time accounting of the mission time tasks. Each row in the C_{NM} matrix corresponds to a target, and each column corresponds to a UAV. When a task of the n-th target is completed by the m-th UAV, the cumulative completion time c_n is calculated by Eq(3).

$$C_{N,M} = \begin{bmatrix} c_{1,1} & \ldots & c_{1,M} \\ \ldots & c_{n,m} & \ldots \\ c_{N,1} & \ldots & c_{N,M} \end{bmatrix}_{N\times M} \tag{2}$$

$$c_{n,m} = \max\{c^{\max}_{nRow}, c^{\max}_{mColumn}\} \tag{3}$$

$$c^{\max}_{nRow} = \max\{c_{n,i} \mid i = 1,2\ldots M\} \tag{4}$$

$$c^{\max}_{mColumn} = t_{i,j} + \max\{c_{j,m} \mid j = 1,2..N\}, t_{i,j} \in T \tag{5}$$

$c^{\max}_{nRow}$ means the new task at the n-th target has to wait until all tasks so far have been completed. $c^{\max}_{mColumn}$ means that after the m-th vehicle completes all its own preceding tasks, it will takes t_{ij} to arrive at the n-th target. The largest value appearing in any cell of matrix $C_{N.M}$ is thus the maximum elapsed time for the mission, which is the maximum time taken by any of the UAVs.

C. Target Sequence Array O_{NK}

The target sequence array O_{NK}, a row vector with integer values, indicates the order that the targets should be visited. There are $N\cdot K$ elements in O_{NK}. For element o_{i-k}, i is the target number and k stands for the time precedence of targets. For example, a scenario with two targets and two tasks is shown in Table 3. The value of the 3rd element is 1, which appears second time in the vector, so the corresponding operation is o_{1-2}, which means the 2nd task at the 1st target.

Table 3: Demonstration of sequence array O_{NK}

O_{NK} dimension	(1)	(2)	(3)	(4)
Target Number	2	1	1	2
Operation	$o_{2\text{-}1}$	$o_{1\text{-}1}$	$o_{1\text{-}2}$	$o_{2\text{-}2}$

D. Assignment Matrix Π_{KN}

The UAV assignment matrix Π_{KN} is a K by N matrix (see Eq.(6)). Each row stands for a task and each column stands for a target (illustrated in Figure 1). π_{KN} is the element in the k-th row and the n-th column of Π_{KN} and stands for the UAV number which performs the k-th task of the n-th target.

$$\Pi_{K,N} = \begin{bmatrix} \pi_{1,1} & \dots & \pi_{1,\ N} \\ \dots & \pi_{k,\ n} & \dots \\ \pi_{K,1} & \dots & \pi_{K,\ N} \end{bmatrix}_{K \times N} \tag{6}$$

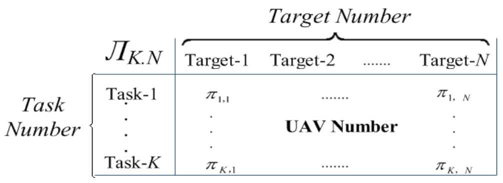


Figure 1: Illustration of UAV Assignment Matrix

III. CONSTRAINTS TREATMENT

Generally, the constraints of UAV assignment problems are derived and represented as equality and inequality equations, in accordance with the requirements of the models and optimization methods. Often used is mixed-integer linear programming (MILP) whose advantages are that the problem and constraints are organized as standard mathematical equations, and the optimal solution can be achieved. On the other hand, this leads to a size explosion of variables with increasing problem complexity. According to the model definition in Eq.(1), and the constraints and data structures in the 2nd section, constraints can be classified into two categories: rigid constraints and soft constraints (see Table 4).

Table 4: Constraint Classification

	Constraints in Table 1
Rigid constraints:	the 3rd, 4th, 5th, 6th, 9th, 10th
Soft constraints:	the 1st, 2nd, 7th, 8th

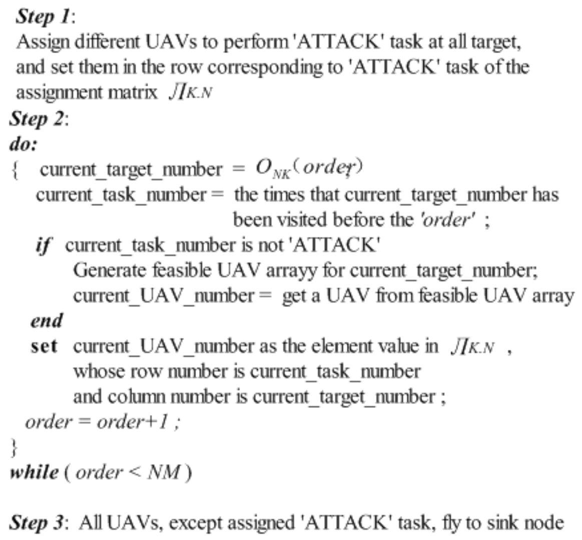
A. Rigid Constraints Treatment

To deal with the rigid constraints, it is necessary to design techniques which will check and adjust potential solutions to meet the constraints' requirements for each iteration. The pseudo-code to generate an assignment matrix is shown in Figure 2. The 5th constraint has the highest priority in the assignment matrix, so vehicles which implement the "ATTACK" task are assigned. According to some selection methods that can be designed many ways, the feasible vehicle is selected from the feasible UAV array. The available vehicle list includes all UAVs which satisfy all constraints at that time and is illustrated in Figure 3.

The 3rd constraint means that the UAV m cannot perform a 'CLASSIFICATION' at another target before the current implementation order in O_{NK} ,, and also be assigned to perform 'ATTACK' at the same target after the current implementation order in O_{NK} .

The 4th constraint in Figure 3 means that a vehicle which has visited a target for 'CLASSIFICATION' or 'ATTACK' is not allowed to be allocated for 'VERIFICATION'.

The 5th constraint: in Figure 3 means that vehicles, performing 'ATTACK' before the current implementation order in O_{NK}, are not feasible to be assigned to other tasks or targets.



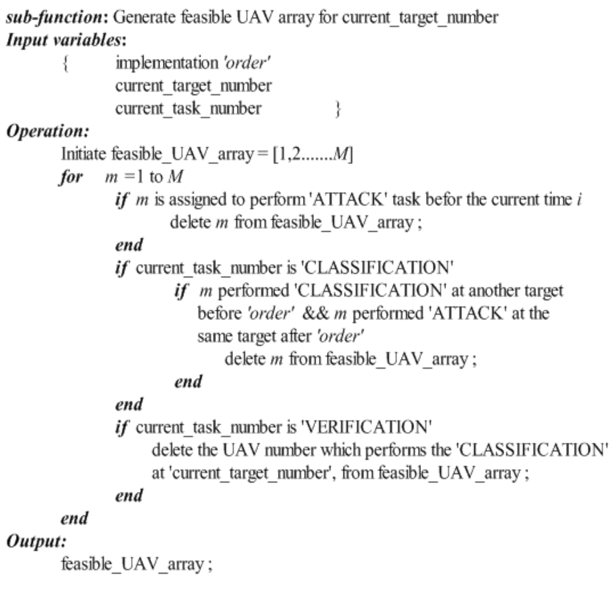
```
Step 1:
  Assign different UAVs to perform 'ATTACK' task at all target,
  and set them in the row corresponding to 'ATTACK' task of the
  assignment matrix $\Pi_{K.N}$
Step 2:
do:
{  current_target_number = $O_{NK}(order)$
   current_task_number = the times that current_target_number has
                         been visited before the 'order' ;
   if current_task_number is not 'ATTACK'
       Generate feasible UAV arrayy for current_target_number;
       current_UAV_number = get a UAV from feasible UAV array
   end
   set current_UAV_number as the element value in $\Pi_{K.N}$ ,
       whose row number is current_task_number
       and column number is current_target_number ;
   order = order+1 ;
}
while ( order < NM )

Step 3: All UAVs, except assigned 'ATTACK' task, fly to sink node
```

Figure 2: Pseudo-code to generate assignment matrix

The 6th constraint means that in Figure **2**, the 'Step 3' operation forces all vehicles, except those allocated to 'ATTACK', to fly to the sink node.

The 9th and10th constraints are guaranteed by Eqs (5) and (4), separately.



```
sub-function: Generate feasible UAV array for current_target_number
Input variables:
     {   implementation 'order'
         current_target_number
         current_task_number          }
Operation:
     Initiate feasible_UAV_array = [1,2.......M]
     for  m =1 to M
         if m is assigned to perform 'ATTACK' task befor the current time i
               delete m from feasible_UAV_array ;
         end
         if current_task_number is 'CLASSIFICATION'
               if m performed 'CLASSIFICATION' at another target
                  before 'order' && m performed 'ATTACK' at the
                  same target after 'order'
                      delete m from feasible_UAV_array ;
               end
         end
         if current_task_number is 'VERIFICATION'
               delete the UAV number which performs the 'CLASSIFICATION'
               at 'current_target_number', from feasible_UAV_array ;
         end
     end
Output:
     feasible_UAV_array ;
```

Figure 3: Sub-function to generate feasible UAV array

B. Soft Constraints Treatment

The soft constraints are satisfied automatically due to the model configuration.

For the 1st constraint, in Eq.(6) and Figure 1 there are only K rows of Π_{KN} which equals the task number and stands for K tasks of a target separately.

For the 2nd constraint, in Eq.(6) and Figure 1, each element of Π_{KN}, corresponds to a UAV number. There is exactly one UAV assigned to perform a task at a target.

For the 7th constraint, in Eqs. (2) and (4), the time-of-flight is accumulated only based on the preceding visited node. It means vehicles cannot fly out of a node, unless it executes a task at the node.

The 8th constraint, in Eq.(5), means that a vehicle either flies to implement another task at a node, or just flies to the sink node directly, which also could be considered as a mission of a vehicle.

IV. PSO APPLICATION

The particle swarm optimization (PSO) algorithm, developed by Kennedy and Eberhart [20], is a population based parallel optimization technique which exhibits good performance in numerous applications.

A. PSO Algorithm

The particle swarm population is initialized by assigning random positions and velocities to each individual. Potential solutions are then flown through the solution space. Each particle keeps track of its "best" information (particle vectors). At time k+1, the update equations of the i-th particle in the d-th dimension of the search space for the standard PSO algorithm are defined as follows:

$$v_{id}^{k+1} = w_i v_{id}^k + c_1 r_{id}\left(p_{id}^k - x_{id}^k\right) + c_2 r_{gd}\left(p_{gd}^k - x_{id}^k\right) \tag{7}$$

$$x_{id}^{k+1} = x_{id}^k + \eta \cdot v_{id}^k \tag{8}$$

which satisfies $|v_{id}^k| \le V_{\max}$, where w_i is the inertia weight, x_{id}^k is the current position of the particle, v_{id}^k is the velocity vector, c_1 and c_2 are acceleration factors, respectively, r_{id} and r_{gd} are random numbers in the range [0,1], p_{id}^k is the personal best position of the particle; p_{gd}^k is the swarm best position among all particles, which is determined by each particle's neighborhood topology.

B. Neighborhood Topology

The neighborhood comprises the particle and a selected number of its topological neighbors [21]. Typical neighborhood topologies include circle, wheel, star, random edges structure, etc. The star structure, one of the most original and popular information topologies, is a full size neighborhood structure such that the best particle in the swarm is selected among all individuals and all particles share the same swarm best information. It is a kind of 'Blackboard' cooperation model, which has strong information exchange pressure and sometimes causes premature convergence. So, in this application, a 2-neighbor neighborhood ring structure is adopted so as to slow down the particle convergence speed and avoid stagnating into local optima as far as possible.

C. Fitness Function

The cumulative time cost is stored in the objective time matrix C_{NM} (Eq.(2)). The largest value appearing in any cell of the matrix is thus the maximum elapsed time for the mission, which is the maximum time taken by any of the UAVs. This time is the fitness value for a given schedule, and is being optimized (minimized) by the PSO algorithm.

D. Encoding the Particle Position Vector

The position vector X of a particle in PSO is a $K\cdot$ $(N+M)$ vector with real values, which consists of two parts. The first $K\cdot N$ elements of X (as shown in Eq.(9)) correspond to the target sequence array (Eq.(10)) and the last $K\cdot M$ variables (as shown in Eq.(11)) stand for the assignment matrix.

$$X = [x_1, \ldots, x_n, \ldots, x_{K\cdot N}, x_{K\cdot N+1}, \ldots, x_v, \ldots, x_{K\cdot N+K\cdot M}] = [X_S, X_A] \tag{9}$$

$$X_O = [x_1 \ldots x_n \ldots x_{K\cdot N}]_{1\times K\cdot N} \tag{10}$$

$$X_\Pi = [x_{K\cdot N+1} \ldots x_\pi \ldots x_{K\cdot N+K\cdot M}]_{1\times K\cdot M} \tag{11}$$

X_O is sorted first and then the modules of these sorted serial numbers are calculated to acquire the target sequence array, explained as Table 5.

Table 5: Target Sequence Array

X_O	→	$x1$	$x2$	$x3$	$x4$
Real value	→	12.315	10.343	55.819	19.556
O_{NK}	→	3	4	1	2

X_Π can be first organized as a K by M matrix which is then encoded to the assignment matrix $\Pi_{K.N}$ following the process in Figure 2 and Figure 3.

V. NUMERICAL EXPERIMENTS

The classic UAV assignment problem in [3] is studied in this section. Various scenarios with different targets, UAVs and tasks requirements are considered. An assumption is made in advance that the time of attack delay and flight time to the sink node are not taken into account. The assumption makes the fitness (Eq.(1)) of scenarios with three tasks the same as those with two tasks , so as to verify the optimization results for both of them.

A. Experimental Design

Considering its ability to avoid most local optima, the local version of PSO is utilized here, since with the increase of the UAV assignment problem size, not only the complexity and dimension of the problem grow rapidly, but also induce many more local optima. The experiment is classified into two categories; one is the scenarios with two tasks and the other with three tasks. Both of them include several sub-scenarios in terms of various numbers of targets and vehicles. The results of applying local PSO will compared with the problem formulated as a Mixed-Integer Linear Programming problem using the GLPK (GNU Linear Programming Kit). All algorithms were implemented in Matlab 7. Parameter settings are given in Table 6.

Table 6: PSO parameter settings

Parameters	Settings
Population size	200
Neighborhood size	2
Inertia weight	0.79
c_1, c_2	2.05

C. Two-task scenarios

In this experiment, only two tasks, 'ATTACK' and 'VERIFICATION', were considered. So the dimension of each particle vector of PSO is 2(N+M). All experiments were run 1,000 times. The format of the 'Scenario' column in Table 7, is [Target, Task, UAV]; elements correspond the numbers of the variables.

Table 7: PSO Experiment on Two Tasks Scenarios

Scenario	Fitness J*	Time Cost sec		Iteration Cost		
		Min	Avg	Min	Avg	Std
[1,2,3]	7.0711	0.072	0.076	1	1	0
[2,2,3]	10.831	0.080	0.080	1	1	0
[3,2,4]	14.316	0.856	0.093	1	1.14	0.535
[4,2,5]	13.831	0.940	0.481	1	9.4	7.567
[5,2,6]	16.099	0.110	3.798	1	75.02	134.7

The average computation time costs by the two methods are compared in Table 8 and Figure 4, which reflect the effect of the size and complexity of the scenarios. Obviously, the computation time cost of both algorithms increases with the change of the number of target and vehicles. Moreover, shown in Table 8, search process of GLPK leads to an unacceptable computation time of the [5,2,6] scenario.

Table 8: Average time cost of two tasks comparison between PSO and GLPK

Scenarios	PSO (sec)	GLPK (sec)
[1,2,3]	0.076	**0.047**
[2,2,3]	**0.080**	0.157
[3,2,4]	**0.093**	2.860
[4,2,5]	**0.481**	194.094
[5,2,6]	**3.798**	More than 6 hours

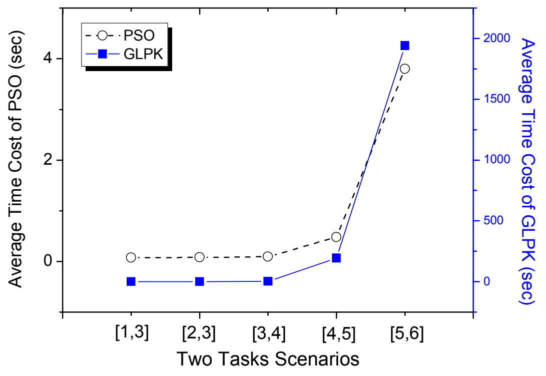


Figure 4: Comparison of two tasks between PSO and GLPK

D. Three-task scenarios

Three-task scenarios are more complicated. UAVs have to complete all three tasks at all targets. Introducing a new task makes the scenario much more complicated compared to the increase of numbers of targets and UAVs, since it not only leads to a large quantity of variables, but also results in more complex constraints.

Table 9: PSO experiment on three-task scenario

Scenario	Fitness J*	Time Cost /sec		Iteration Cost		
		Min	Avg	Min	Avg	Std
[1,3,3]	7.0711	1.24	1.26	1	1	0
[2,3,3]	10.831	1.52	1.54	1	1	0
[3,3,4]	14.316	1.83	1.84	1	1.29	0.782
[4,3,5]	13.831	2.12	22.41	1	20.68	14.08
[5,3,6]	16.099	34.44	410.14	28	348.43	288.70

The dimension of each particle vector of PSO is 3(N+M). The population size and parameter settings are the same as for the previous experiment. The numerical results are presented in Table 9 and Table 10. Compared to the two-task scenarios, it takes much more computation time to complete the assignment mission plan. GLPK exhibits better performance when dealing with very small size problems (the first two scenarios in Table 9). However, for realistic scenarios such as are encountered in operational settings, GLPK cannot achieve the optimization within an acceptable time.

Table 10 : Average time cost of three-task scenario comparison between PSO and GLPK

Scenarios	PSO (sec)	GLPK (sec)
[1,3,3]	1.26	**0.062**
[2,3,3]	1.54	**0.425**
[3,3,4]	**1.84**	467.282
[4,3,5]	**22.41**	More than 6 hours
[5,3,6]	**410.14**	More than one day

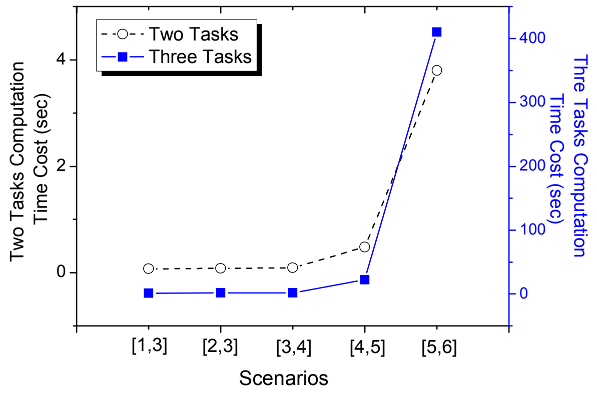


Figure 5: Comparison of PSO computation times between two-task and three-task scenarios

E. Impact of the Number of Tasks

The consideration of computational complexity and time cost becomes more important with realistic operational scenarios. In Figure 5, the computation time cost of PSO is compared between two-task and three-task scenarios. The unit of both left and right vertical coordinates is seconds. An interesting result in the figure is that the computation time cost of PSO increases almost 1000 times when there is a new task added.

F. Effects of PSO Neighborhood Size

In PSO, with more neighbors a particle has a higher probability of interacting with others and can sufficiently benefit from the best information in the swarm. However, there is a potential risk that it will lead to individuals that become more similar, and thus it is easier to stagnate into a local optimum. In Table 11 there is a comparison of the average computation times of PSOs with various neighborhood sizes. All experiments are run 100 times and only the results which achieve the optimum are taken into account in the table. It shows that the PSO with two neighbors on each side (four total) is not as fast as others that have more neighbors in finding an optimum. However, with the increase of the problem complexities, not all neighborhood sizes can guarantee a 100 percent success rate.

In Figure 6, only the PSOs with neighborhoods of 2 and 5 can complete all the two-task scenario optimizations. In Table 12 it is noteworthy that the 2-neighbor version of PSO beats the 5-neighbor version for three-task scenarios.

Table 11: Two tasks scenarios comparison among PSOs with different neighbor size

Neighbor	*Average time cost (sec)*				
size	[1,2,3]	[2,2,3]	[3,2,4]	[4,2,5]	[5,2,6]
2	0.076	0.080	0.093	0.481	3.798
5	0.072	0.077	0.088	0.326	1.741
10	0.073	0.077	0.086	0.302	1.692
50	0.075	0.076	0.086	0.350	1.526
100	0.076	0.082	0.095	0.364	1.183
Full	0.074	0.078	0.097	0.543	1.168

Table 12: Success rate comparison between 2-Neighbor and 5-Neighbor PSO algorithms

Scenario	*Pop Size*	*Neighbor*	*Runs*	*Success Rate*
[5,3,6]	200	2	10,000	100
[5,3,6]	200	5	100	65

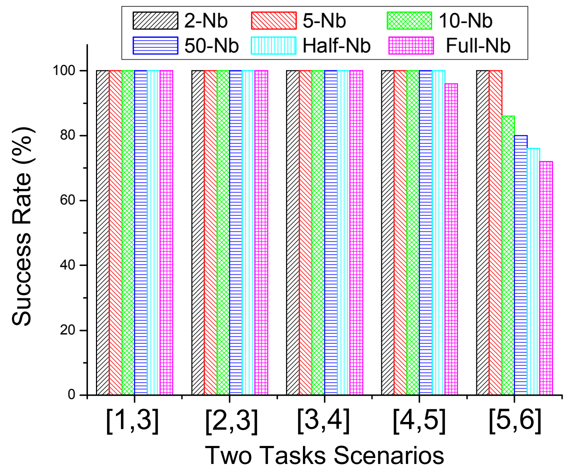


Figure 6: Rate comparison among various neighborhood sizes. (Nb = Neighborhood)

VI. DISCUSSION

The UAV assignment model, proposed in the paper, can be easily adapted by any computation intelligence algorithms and also reduce the dimension of the solution space. Furthermore, there are two advantage of the proposed encoding method in the report. First, real value number is utilized so that it is very easy to realize and calculate. Second, since it is based on the sort of particle vector to transfer real value vector to integral sequence array and assignment matrix, there is no any value limitation and initial requirement of the particle position vector.

A 2-neighbor version of PSO is applied in various scenarios to assign vehicles in two-task or three-task scenarios. Numerical experiments are conducted to illustrate the performance compared to MILP methods. The next steps are to refine the model for more complicate scenarios and improve algorithm's flexibility and performance for more tasks.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of Air Force Research Laboratory Contract FA9550-07-C-0167, and the State of Indiana's 21st Century Research and Technology Fund. Special thanks go to Dr. Corey Schumacher and his colleagues at Wright Patterson Air Force Base for their assistance and collaboration.

VIII. REFERENCES

[1] P. Chandler, M. Pachter, D. Swaroop, and J. Fowler, "Complexity in UAV cooperative control," in *ACC*, 2002, pp. 1831-1836.

[2] L. Tao, "Research on Distributed Task Allocation and Coordination for Multiple UCAVs Cooperative Mission Control," in *Control Science and Engineering*. vol. PHD ChangSha: National University of Defense Technology,, 2006.

[3] C. Schumacher, P. R. Chandler, M. Pachter, and L. S. Pachter, "Optimization of Air Vehicle Operations Using Mixed-Integer Linear Programming," Air Force Research Lab (AFRL/VACA) Wright-Patterson AFB, OH Control Theory Optimization Branch 2006.

[4] C. Schumacher, P. Chandler, and L. Pachter, "UAV task assignment with timing constraints via mixed-integer linear programming," in *Proc. AIAA 3rd Unmanned Unlimited Systems Conference*, 2004.

[5] M. Alighanbari, Y. Kuwata, and J. P. How, "Coordination and control of multiple UAVs with timing constraints and loitering," in *Proceedings of the 2003 American Control Conference* 2003, pp. 5311-5316.

[6] W. Guo, K. E. Nygard, and A. Kamel, "Combinatorial Trading Mechanism for Task Allocation," in *Proceedings of the 14th International Conference on Computer Applications in Industry and Engineering*, Las Vegas, Nevada, USA, 2001.

[7] A. Arulselvan, C. W. Commander, and P. M. Pardalos, "A hybrid genetic algorithm for the target visitation problem," 2008.

[8] B. R. Secrest, "Traveling salesman problem for surveillance mission using particle swarm optimization," in *School of Engineering and Management of the Air Force Institute of Technology*. Master's thesis: Air University, 2001.

[9] K. S. Vijay, S. Moises, and N. Rakesh, "Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles." vol. 35: Elsevier Science Ltd., 2008, pp. 1813-1828.

[10] R. A. Flood, "A JAVA Based Human Computer Interface for a UAV Decision Support Tool Using Conformal Mapping," Wright-Patterson AFB OH: Air Force Institute of Technology (AU), 1999.

[11] K. P. O'Rourke, T. G. Bailey, R. Hill, and W. B. Carlton, "Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search," *Military Operations Research Journal,* vol. 6, 2000.

[12] K. E. Nygard, P. R. Chandler, and M. Pachter, "Dynamic network flow optimization models for air vehicle resource allocation," in *Proc. of the the American Control Conference*, Arlington, Texas, 2001, p. 1853~1858.

[13] C. Schumacher, P. R. Chandler, S. J. Rasmussen, and D. A. W. D. Walker, "Task allocation for wide area search munitions with variable path length," in *Proceedings of the 2003 American Control Conference*, 2003, pp. 3472-3477.

[14] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly,* vol. 2, pp. 83-97, 1955.

[15] K. A. Darryl, H. B. Arnold, and R. John, "Assignment scheduling capability for unmanned aerial vehicles: a discrete event simulation with optimization in the loop approach to solving a scheduling problem," in *Proceedings of the 38th conference on Winter simulation* Monterey, California: Winter Simulation Conference, 2006, p. 1349~1356.

[16] S. J. Rasmussen and T. Shima, "Tree search algorithm for assigning cooperating UAVs to multiple tasks," *International Journal of Robust and Nonlinear Control,* vol. 18, p. 135~153, 2007.

[17] G. Chen, J. Jose, and B. Cruz, "Genetic algorithm for task allocation in UAV cooperative control," in *Proc. AIAA Guidance, Navigation and Control Conference*, 2003.

[18] S. Tal, J. R. Steven, G. S. Andrew, and M. P. Kevin, "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms." vol. 33: Elsevier Science Ltd., 2006, pp. 3252-3269.

[19] e. a. Hai-Bin DUAN, "Methods of multi-UAVs' mission assignments based on basic ant colony intelligence," in *National Invention Patents*. vol. 200710121762.7 P.R.China, 2007.

[20] J. Kennedy and R.C.Eberhart. Particle swarm optimization. *Proc. IEEE International Conference on Neural Networks,* vol. VI, pp. 1942-1948, 1995.

[21] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.* , 1999, pp. 1931-1938.