

## Chapter 1

# AN INTEGER PROGRAMMING MODEL FOR ASSIGNING UNMANNED AIR VEHICLES TO TASKS

Michael J. Hennebry

*Department of Computer Science and Operations Research  
North Dakota State University  
Fargo, ND 58105-5164*

Ahmed Kamel

*Department of Computer Science and Operations Research  
North Dakota State University  
Fargo, ND 58105-5164*

Kendall E. Nygard

*Department of Computer Science and Operations Research  
North Dakota State University  
Fargo, ND 58105-5164  
Kendall.Nygard@ndsu.nodak.edu*

**Abstract** We consider a team of unmanned air vehicles (UAVs), deployed to carry out tasks such as searching for, classifying, attacking, and performing battle damage assessment of enemy targets. In some missions, it may be advantageous for the UAVs to form sub-teams to leverage their efforts and cooperatively handle single enemy targets. Within a sub-team, each UAV would be responsible for carrying out a specific set of tasks to collectively accomplish the mission. An integer linear program is posed for assembling the vehicles into sub-teams. The model's novelty and usefulness lie in its combination of flexibility and ease of solution. Empirical work suggests that good solutions can be computationally generated rapidly enough to assimilate new situation awareness information dynamically, and formulate new sub-teams on the fly.

**Keywords:** Integer programming, team formation, unmanned air vehicles

## Introduction

The UAV missions of interest use a team of UAVs with incomplete a priori information regarding potential targets and their locations. Each UAV is equipped to search for enemy units, classify those units, attack, and carry out battle damage assessment. In some situations, it is desirable for several UAVs to form sub-teams to enhance their effectiveness. Examples include: i) using multiple UAVs to simultaneously strike a single target to gain the element of surprise and maximize the probability of destroying it, ii) deliberately utilizing a low-cost UAV as a decoy to induce an enemy radar to reveal itself, followed by a strike carried out by another, and iii) utilizing one UAV to classify and to attack, followed by another to assess damage. The model is focused on the sub-team formulation decisions, and is intended to be rerun each time that new information might make reformulation of sub-teams desirable. Such information would include such things as pop-up or disappearing targets or threats and battle damage assessment reports. Ideally each UAV would have the same information and could run the model and get the same answer as the other UAVs, accomplishing coordination without a central authority. The model can be viewed as a generalization of a linear network model presented in [Nygard et. al.], which is limited to the assignment of a single UAV per target.

## 1. MODEL

In the model, each prospective team is distinguished by its target and its roles. Forming a team requires filling each of its roles with a distinct UAV. In this exposition,  $t$ ,  $u$ ,  $g$ , and  $r$  index targets, UAVs, teams, and roles within teams respectively. All these indices start at one.

This model operates on the following input data:

$vals_u$	is the value of having UAV $u$ search.
$target_g$	is the target handled by team $g$ .
$numRoles_g$	is the number of roles in team $g$ .
$valt_{g,r,u}$	is the value of assigning UAV $u$ to team $g$ in role $r$ .

This model solves for the following zero-one decision variables:

$search_u = 1$	iff UAV $u$ is assigned searching.
$formed_g = 1$	iff team $g$ is formed.
$teamed_{g,r,u} = 1$	iff UAV $u$ is assigned role $r$ within team $g$ .

The index set of the variable  $teamed$  need not be a complete cartesian product, since teams are not required to have the same number of roles, and UAVs are not required to be assignable to all available roles.  $valt_{g,r,u}$  exists if the corresponding assignment is allowed.

The model requires solving the following integer program.

Maximize

$$\sum_u \text{vals}_u \text{search}_u + \sum_{g,r,u} \text{valt}_{g,r,u} \text{teamed}_{g,r,u} \quad (1)$$

subject to:

$$\sum_{\text{target}_g=t} \text{formed}_g \leq 1 \quad \text{for each } t \quad (2)$$

$$\sum_u \text{teamed}_{g,r,u} = \text{formed}_g \quad \text{for each } g, r \quad (3)$$

$$\sum_{g,r} \text{teamed}_{g,r,u} + \text{search}_u = 1 \quad \text{for each } u \quad (4)$$

$$\text{search}, \text{formed}, \text{teamed} \text{ are binary} \quad (5)$$

The objective function provides a linear measure of the total performance value. Constraints (2) specify that each target is handled by at most one team. Constraints (3) specify that if a team is formed, then all its roles must be filled, and if it is not formed, then none of its roles are filled. Equations (4) specify that each UAV does precisely one job. The binary conditions ensure that the variables represent definite decisions.

## 2. MODEL CHARACTERISTICS

The model allows consideration of an exponentially large number of combinations of UAVs being formed into teams without having to explicitly enumerate them. This is accomplished by using linear functions involving only one UAV each in the objective function. The restriction is mitigated considerably by distinguishing the roles within the teams and by not allowing all UAVs in all roles. However, if it was desirable to enumerate each combination of UAVs for comprising target teams, this could easily be done by restricting each role on each team to a specific single UAV. A potentially better tactic would be to group UAVs according to how they fill a particular role. For example, for a simultaneous multi-point attack, one might select UAVs for roles based on the time intervals in which they could arrive at a particular target. Such time considerations also apply to sequential attacks. Also, in a hazardous environment, the value of assigning a UAV to fill a role depends upon the likelihood that it can survive to fill that role.

### 3. SOLUTION METHOD

If the “formed” variables are given fixed values, the resulting problem is a capacitated transshipment problem with all integer constraint data. In this situation, the total unimodularity of every basis matrix guarantees integral solutions. The model can be relaxed to make the “formed” variables non-negative integers and the other variables non-negative reals. We used the branch and bound MILP (mixed integer linear program) solver in GLPK (GNU Linear Programming Kit) [Makhorin] to solve the problem directly. To potentially improve solution times for large problems, we also developed a heuristic procedure. At each iteration of this procedure, a specified subset of the integer variables is forced to be integer. Once their values were obtained, they are fixed for subsequent iterations. The heuristic is parameterized by “chunkSize”, the number of teams whose formed variables are required to be integer at each iteration. The solver uses the following procedure:

```

relax integrality condition
Solution = solve( )
while Solution not all integral {
    choose chunkSize targets, starting with
        those that have a fractional “formed” variable
    update the problem to require the “formed” variables of
        the chosen targets to be integral
    Solution = solve( )
    update problem to fix integer variables at their Solution values
}

```

At any iteration, each target is given a score that is the maximum value of any of its “formed” variables. The score of variable  $\text{formed}_g$  is  $\min(\text{formed}_g, 1 - \text{formed}_g)$ . The chosen targets are the ones with the largest scores.

### 4. COMPUTATIONAL EXPERIMENTS

The test problems came from a problem generator in which parameters were selected from a range using a pseudo-random value drawn from a uniform distribution. The generator proceeds as follows:

- Places the given number of targets on the 100x100 square.
- Assigns each target an intrinsic value in a given range.
- Assigns each target a minimum team size in the range 1 to the given maximum.
- Places the given number of UAVs on the 100x100 square.

- Assigns a value in the given range, to each UAV's search.
- Distributes the given number of teams among the targets.
- Distributes the target value among the roles in each team.
- Generates the heading associated with each role.
- Computes the value of each UAV filling each role, selecting only the given number of best UAVs.

Whenever possible, the problem generator distributes teams uniformly among the targets. Otherwise the more valuable targets receive one more team than the less valuable targets.

To distribute target values among roles, the generator gives each role a value in proportion to an integer in the range 30..129.

The generator selects the heading of the first role of each team, then disputes the others evenly around the compass.

The generator calculated the value of each UAV filling a role by multiplying the value of the role by a discount factor that is a function of the distance between the UAV and the target and on the angle formed by the headings of the role and the UAV. The computation is given by:

$$\text{value} = \text{rint} \left( \frac{\text{roleValue}}{1 + (\text{distance} + \text{wap})/\text{halfRange}} \right) \quad (6)$$

where

$$\text{wap} = \text{wapMax} * (1 - \cos(\text{uavHeading} - \text{roleHeading})) / 2 \quad (7)$$

Note that if there is no wrong angle penalty (wap), the distance halfRange is the distance at which a role loses half its value. The function rint rounds to a nearest integer.

The experiments were run under the Linux operating system on a computer with a 866.398 MHz pentium 3 processor.

Each datum for objective, % error and cpu seconds is based on runs on 100 distinct problems, each with 16 UAVs. Each test problem was solved to optimality as well as with the heuristic. The heuristic was run with various settings for the chunkSize parameter. In Table 1.1, a chunk size of "opt" corresponds to the optimal solution. Average percentage error and standard deviations are reported in each case. The "num slow" column gives the number of runs which took more than twice as long as average. The "cpu sec w/o tails" data is for the middle 90 runs for each chunk size. A tail size of 5 was chosen as  $\lfloor \sqrt{N/3} \rfloor$  with  $N = 100$ . The formula was chosen to be roughly proportional to  $\sqrt{N}$  and to give the correct answers, 0, 0, and 1, respectively, for  $N=1$ ,  $N=2$ , and  $N=3$ . The error of the heuristic was consistently small and is arguably close

enough to the optimum for practical use, particularly since there are likely to be inaccuracies in the data in a real situation. We conclude that the heuristic demonstrates that good solutions to the sub-team formation problem can be quickly and accurately generated.

Table 1.1. Computational Results

num targets	num teams	chunk size	average objective	% error		cpu seconds		num slow	cpu w/o tails	
				av	SD	av	SD		av	SD
20	120	opt	829	0.00	0.00	12.0	24.7	12	7.6	12.7
		1	827	0.26	0.49	4.0	3.0	5	3.7	1.9
		2	828	0.16	0.36	3.6	2.6	8	3.3	1.7
		5	827	0.17	0.58	5.1	5.3	11	4.4	3.2
		10	828	0.12	0.31	6.9	9.8	12	5.3	5.1
		20	829	0.00	0.00	12.0	24.5	12	7.7	12.7
		50	829	0.00	0.00	12.0	24.8	12	7.7	12.8
30	180	opt	898	0.00	0.00	47.1	172.1	9	20.3	29.0
		1	894	0.46	0.67	10.0	7.5	8	9.2	4.6
		2	896	0.31	0.54	8.6	7.0	10	7.9	4.2
		5	897	0.17	0.36	12.8	15.5	10	10.5	7.9
		10	897	0.16	0.38	27.8	97.7	8	13.6	14.4
		20	897	0.10	0.35	38.2	160.6	6	16.3	18.2
		50	898	0.00	0.00	47.1	172.2	9	20.3	29.0
40	240	opt	929	0.00	0.00	152.0	457.9	11	71.6	153.1
		1	925	0.43	0.70	14.7	10.3	8	13.7	7.1
		2	926	0.36	0.60	13.5	10.9	6	12.1	6.3
		5	926	0.34	0.65	27.5	45.1	13	19.9	18.1
		10	927	0.26	0.54	51.6	124.7	12	29.8	47.1
		20	928	0.15	0.36	66.7	169.6	11	33.0	56.3
		50	929	0.00	0.00	152.2	456.0	11	72.2	155.1
50	300	opt	958	0.00	0.00	314.5	749.0	14	179.0	298.1
		1	952	0.63	0.97	25.5	16.8	8	23.9	11.7
		2	953	0.55	0.81	21.1	12.1	8	20.3	9.5
		5	953	0.52	1.00	36.3	35.7	11	31.5	22.4
		10	953	0.56	1.05	78.0	155.0	10	50.1	57.5
		20	953	0.51	0.90	116.0	262.6	10	67.1	87.8
		50	958	0.00	0.00	313.7	745.6	14	179.2	299.2
60	360	opt	972	0.00	0.00	365.3	1217.0	10	160.8	302.3
		1	967	0.47	0.75	32.8	24.5	7	30.2	14.5
		2	967	0.44	0.70	27.8	26.3	5	24.9	11.5
		5	968	0.35	0.58	38.8	31.8	13	35.6	23.9
		10	968	0.39	0.70	94.5	244.4	11	58.3	75.9
		20	969	0.26	0.49	117.5	347.3	11	62.6	84.3
		50	970	0.22	0.58	285.5	1108.5	10	120.0	210.3

**References**

- Chandler, P. R., M. Pachter, D. Swaroop, J. Fowler, J. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard. (2002). "Complexity in UAV Cooperative Control", Anchorage, AK: *Proceedings of the 2002 American Control Conference*
- Makhorin, Andrew. (2001). *GNU Linear Programming Kit Version 3.0 User's Guide*, <http://www.gnu.org/software/glpk/>
- Nygaard, K. E., P. R. Chandler, and M. Pachter. (2001). "Dynamic Network Flow Optimization Models for Air Vehicle Resource Allocation," Arlington, VA: *Proceedings of the 2001 American Control Conference*