A Goal Programming Approach to the Team Formation Problem

Fang Liang [†]
Stephen Lawrence [‡]

Leeds School of Business
University of Colorado
419 UCB
Boulder, CO 80309-0419

ABSTRACT

The team formation problem consists of forming a work team from a large collection of candidates with disparate skills and attributes. The team formation problem is ubiquitous in practice, for example, product development teams formed from marketing, engineering, and finance skills; software development team formed from programmers and software engineers with needed programming and systems skills; and construction management teams formed from architects, civil engineers, designers, and construction engineers. The generally used methods to form team are random assignment, self-selection and facilitator assignment.

We formulate the team formation problem as a mixed integer linear goal program. The use of goal programming is appropriate in this context because the constraints of the team formation problem are usually soft or fungible, and can be traded off against one another depending on their priority. Our formulation also allows for the inclusion of "hard" or categorical constraints. Since the team formation problem is known to be NP-Hard, we develop a heuristic solution methodology to rapidly find good solutions to the problem. We adapt the Greedy Randomized Search Procedure (GRASP) to the team formation problem and test on a variety of problems. In addition, we employed standard IP solution software (CPLEX) to solve our set of test problems to optimality, where possible.

Preliminary results indicate excellent performance for the GRASP method in this context. For the problems tested, GRASP provided the optimal solution wherever an optimal solution could be found. In several cases for reasonably sized problems (50 team members) an optimal solution could not be identified even after 1 hour of computation. In contrast, the GRASP method found its solutions in less than 3 seconds, even for the largest problems. Future research will test a larger, more complex set of problems, and will verify the effectiveness of the GRASP method in solving the team formation problem across a large range of problem settings.

[†] Fang.Liang@colorado.edu
[‡] Stephen.Lawrence@colorado.edu

Introduction

The team formation problem consists of forming a work team from a large collection of candidates with disparate skills and attributes. The team formation problem is ubiquitous in practice, for example, product development teams formed from marketing, engineering, and finance skills; software development team formed from programmers and software engineers with needed programming and systems skills; and construction management teams formed from architects, civil engineers, designers, and construction engineers. The generally used methods to form team are random assignment, self-selection and facilitator assignment. (See [1])

Random assignment is often chosen because it is easy to use and it looks fair. But it is not an ideal method especially when multi-criteria constraints need to be considered, such as gender, age and working experience, it is impossible using a random method to achieve a desirable objective. Self-selected team works better on average because the members may feel more ownership of team problems. But it only happens when the members know each other well, and again, it is difficult to satisfy several constraints simultaneously. Facilitator-assigned method is our concern. Usually the facilitators have several criteria to make the team balanced depending on their experience. The central issue is how to form teams in an effective way, satisfying all these criteria simultaneously. Our goal is to try to use mathematical method to get the optimal solution of the team formation problem.

Mathematical Formulation

The team formation problem is an extension of the classic assignment problem:

$$\text{Min } \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{s.t. } \sum_i x_{ij} = 1, \text{ for every } j$$

$$\sum_j x_{ij} = 1, \text{ for every } i$$

$$x_{ij} = 1, \text{ or } 0$$

where $x_{ij} = 1$ if member i is assigned to job j, and c_{ij} is the cost (or benefit) of assigning member i to job j.

Now we extend the assignment problem by assigning n personnel to m teams (n>m) according to different personnel skill sets. In most settings, team skill requirements are soft or fungible between teams without sharp constraints. In a sense the problem is more a "constraint satisfaction" problem than an optimization problem. While the classic assignment problem is *totally unimodular* (binary solutions guaranteed), the team formation problem is not so we must formulate it as an integer program. Consequently, we can formulate the program as a mixed integer linear goal programming(MILGP). Goal programming allows for very flexible formulation and

allows the specification of priorities among goals. For example, goals might include having at least m_n members on a team, having no more than q engineers on the team, having a mix of certain skills on the team, and so forth. Each of these goals can be assigned different deviation penalty, which allows trade-offs between goals in complex problem settings. The goal programming can be divided into two methods: preemptive goal and not-preemptive goal according to the importance of the goals. Our paper only focuses on the first one.

In non-preemptive goal programming, all goals are of roughly comparable importance. Some objectives may be required to be as close to their goal as possible, referred to as two-sided goals. Some objectives may need to be achieved above or below their respective goals, called upper one-sided goals or lower one-sided goals respectively. (See [2])

Suppose objective i needs to be maintained at its target level f_i^*. The index set of all such objectives is given by:

$I^{=} = \{i \mid f_i(x)$ needs to be achieved as closely to f_i^* as possible, $i=1,\ldots n\}$

Suppose objective j needs to be achieved above its goal level f_j^*. The index set of all such objectives is given by:

$I^{\geq} = \{j \mid f_j(x)$ could be above f_j^* as far as possible, $j=1,\ldots n\}$

Suppose objective k needs to be achieved below its goal level f_k^*. The index set of all such objectives is given by:

$I^{\leq} = \{k \mid f_k(x)$ could be below f_k^* as far as possible, $k=1,\ldots n\}$

Let d_i^+ be a deviation variable, representing the distance that $f_i(x)$ over-achieves f_i^* and d_i^- the distance that $f_i(x)$ under-achieves f_i^*.

Then the goal programming model could be formulated as follows:

$$\text{Min} \quad Z = \sum_{i \in I^{=}} (w_i d_i^+ + w_j d_j^-) + \sum_{j \in I^{\geq}} w_j d_j^- + \sum_{k \in I^{\leq}} w_k d_k^+$$

St. $d_i^+ - d_i^- = f_i(x) - f_i^*$

$Ax = b$

$d_i^+, d_i^- \geq 0 \qquad (i= 1,...n)$

Depending on the above foundations, we use the MBA team formation problem as our test bed and formulate it as a mixed integer linear goal program. In this problem, it consists of assigning MBA students to teams in such a way that all the teams meet a set of specified constrains, such as gender diversity, ethic diversity, test scores, working background and so forth. Our goal is to construct balanced teams that each team satisfies the goals as much as it can. In other words, we want to minimize the penalty of under-achieving or over-achieving the perspective goal values of each team.

We use the following notation to design the teams:

n: number of students

p: number of teams

q: number of attributes associated with each student

m: number of students in each team (we assume m=n/p is an integer number)

a_{ik}: value of attribute k (k=1,…q) for student i (i=1,…n)

w_k: penalty of attribute k (k=1,…q)

b_k: target value of attribute k (k=1,…q). Where b_k is given by the school authority.

d_{jk}^+ : deviation variable to measure the distance that v_{jk} over-achieves the target value b_k and d_{jk}^- the distance that v_{jk} under-achieves b_k.

x_{ij}: binary variable, $x_{ij} = 1$ if student i is assigned to team j; otherwise it is equal to zero.

Then the problem can be formulated as follows:

$$\min \sum_{j=1}^{p} \{ \sum_{k \in I^=} (w_k d_{jk}^+ + w_k d_{jk}^-) + \sum_{k \in I^\geq} w_k d_{jk}^- + \sum_{k \in I^\leq} w_k d_{jk}^+ \}$$

$$\text{St.} \quad \sum_{i=1}^{n} a_{ik} * x_{ij} - (d_{jk}^+ - d_{jk}^-) = b_k \, (j = 1,...p, k = 1,...q)$$

$$\sum_{i=1}^{n} x_{ij} = m (j = 1,...p)$$

$$\sum_{j=1}^{p} x_{ij} = 1 (i = 1,...n)$$

$$d_{jk}^+, d_{jk}^- \geq 0$$

Solution Methods

We can use CPLEX to solve the mixed integer goal programming and get the exact optimal solution. For the problem with p teams and m-student in each team, there are p*m binary variables x_{ij} and 2p deviation variables d_{jk}^+ and d_{jk}^- . Also, if there are q goals relating to students' attributes, for all the team, there will be q*p constraints need to be satisfied. As the number of p, m and q increase, it will dramatically slow down the CPLEX to solve the problem. Therefore, for the large problem with large number of members and teams or large number of goals, we also develop a heuristic approach GRASP (Greedy Randomized adaptive Search Procedures) to solve the team formation problem.

A GRASP is an iterative process, with each GRASP iteration consisting of two phases, a construction phase and a local search phase.(See [5]) In the construction phase, a

feasible solution is iteratively constructed, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all elements in a candidate list with respect to a greedy function, which measures the benefit of selecting each element. Then randomly choose one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the restricted candidate list (RCL). The solutions generated by a GRASP construction are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Therefore, a local search is used to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood.

As for the team formation problem, during the construction phase of GRASP, a member is chosen at a time randomly from the RCL determined by the adaptive greedy function. Since the objective function is to minimize the penalty for all the attribute values, the greedy choice is to select the member that makes all the attribute values of his team has the smallest penalty so far.

For every complete solution, the interchange of 2 member pairs is used for the perturbation of an incumbent solution. Then we can force all members from a randomly chosen team to change their team in the best possible way, i.e., the objective function get a minimum value. Therefore, the local search phase is finished.

The procedure is as follows:

Define the problem and reading data from the file.

For (Grasp stopping criterion not satisfied)

{

a) Randomly choose a member to be in the team 1

b) Calculate the greedy function value for every other member is in this team. Then set up the list of best candidates (RCL= α * n, $0 \leq \alpha \leq 1$). Thus a member is randomly chosen from the list.

where the greedy function $= \sum_j \sum_k w_k \, d_{jk}$ (d_{jk} could be d_{jk}^+ or d_{jk}^- depending on the constraint is defined as <=, >= or =).

c) Go on the step b) to select the next member until the number of the team is satisfied.

d) Randomly choose a member (except the members that have been selected) to be in the next team.

e) Repeat step b) and c) until the team number is satisfied. Save it as the incumbent solution.

f) After getting a complete solution, improve the solution by interchange of 2 member pairs. We calculate the objective function for every possible change of all members from a randomly chosen team with other teams. If the new

solution is better than incumbent, set incumbent = new solution.

g) Repeat a) through f) to generate new solution.

h) If new solution is better than incumbent, set incumbent = new solution.

}

We can see there are two parameters in the searching process. One is α, which balance the solution search between randomized procedure and greedy procedure. When α is close to 1, the list of best candidates almost includes all the members. Thus the choice of next member could be more diversified and randomized. On the other hand, if α is close to 0, it means the next member has seldom choice and the procedure is more greedy. (α=0 is purely random while α=1 is purely greedy.) Usually the GRASP with α=0.8 performs best. The other parameter is the stopping criterion not satisfied. It could be maximum number of iterations, solution sought has been found, or the maximum running time, etc. For different data set, we can adjust α and the stopping criterion to make the search process more efficient.

Symmetries Elimination

There are always symmetries problems during the team formation process that makes the solution methods low efficient in some degree. We mainly consider two kinds of symmetries that need to be eliminated. Firstly, suppose we have assigned all members to p teams and got an optimal solution. If we exchange all the members in group i with the all in group j, then we get a new solution while the objective function does not change. For example, students 0 and 1 in team 1, 2 and 3 in team 2, 4 and 5 in team 3 are an optimal solution. Then assigning 0 and 1 to team 2, 2 and 3 to team 3, 4 and 5 to team 1 is also optimal. So for the p-team problem, there are p! permutations of assigning students to different teams but have the same objective function. Secondly, if the attributes of student i is totally identical with those of student j, then swapping student i and j will make no difference to the objective function. So for the problem with k students having identical attributes, there will be k! permutations accordingly.

The symmetries problem makes the solution time of CPLEX much longer. For example, when we get the first of those optimal solutions and make it the incumbent, but the lower bounds are not perfectly tight, so the other permutations continue to be viable nodes. Because they all have the same objective value as the incumbent, we cannot fathom them on bound. Therefore the solver is doomed to rediscover the same solution over and over again, wasting a lot of time. As for the GRASP, the running time is really fast, so the main effect of symmetries problem on time issue is not a big deal. But if we run the code for several times, we will get different assignment with the same objective function due to this kind of problem. In reality, one possible way of assignment is enough. Therefore, we need to clean up our solution not only for CPLEX but also GRASP.

I offer three possible ways to reduce the amount of symmetries in the problem. The first method is to force the teams to be listed in increasing order of their objective

contributions. In other words, the total penalty for goal deviations in the first team is less or equal than the total penalty for the second team, and so on. This method eliminates permutations of any assignment if the assignment to each team has distinct total penalty for different teams, although permutations will still exist if the teams are tied for it. It introduces fairly small number (p-1) of additional constraints. The second method requires that the groups be filled in lexicographic order. Recalling that x_{ij} =1 if student i is in group j, otherwise it is zero. Lexicographic order means that the summation of x_{ij} for group j must be a larger number than the summation of $x_{i(j+1)}$ for group (j+1). In general, we cannot assign student i to group j unless some student i' < i has been assigned to group (j – 1). This method adds (p - 2)*(p - 2) constraints. Although it slows down the solver quite a bit by much more constraints, it completely precludes any permutations of the team labels. However, the first two methods do not prevent swapping two identical students. The third method is to make the teams be indexed in increasing order of the total identity number of students. This method assumes that each student has an identity number ranged from 1 to n. So when adding the identity number of all the members in group j, it should be a less or equal number than that of group (j+1). This method only adds small number (p-1) of additional constraints, and it eliminates not only the permutation of the team labels but also the symmetries due to the identical attribute of different students. Also, permutations will still be there if the teams are tied for the identity number.

There is a trade-off between the symmetries elimination and the number of extra constraints added. We tested these three methods with different data sets; usually the third method is more time-saving by using CPLEX.

Results

No.	n	p	q	CPLEX solution	GRASP solution	solution gap	CPLEX time	GRASP time	time gap

Reference

[1]. Donald R.Bacon, Kim A.Stewart and Elizabeth Scott Anderson. Methods of assigning players to teams: A review and novel approach. Simulation & Gaming, Vol.32 No.1, March 2001 6-17

[2]. Hillier and Lieberman. Introduction to operations research. 7th edition. Chapter 7.5

[3]. Minimax goal programming for managerial decision making

[4]. Jacques Desrosiers, Nenad Mladenovic and Daniel Villeneuve. Design of balanced MBA student Teams. MIC'2001- 4th Metaheuristics International Conference

[5]. Thomas A.Feo and Mauricio G.C.Resende, Greedy Randomized Adaptive Search Procedures.