The Karate Union of Great Britain (KUGB) hold national championships each year at Crystal Palace, London. The team event consists of five-person teams in a standard knockout tournament. At each stage of the tournament a team must decide in which order the players must compete. They must do this in absence of knowledge of the opposing team’s chosen order and with a view to optimizing the total expected final score of the team. It is apparent that some thought is put into this decision and the purpose of this article is to consider if OR has something to offer in the way of advice.

We begin by making the strenuous assumption that, although we do not know the order of play of the opposing team, we have information regarding the relative strengths of each pair of players. This information may be encoded in a skill matrix as follows.

<table>
<thead>
<tr>
<th>Team A</th>
<th>Team B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

A typical encounter may be described as A{1,2,3,4,5} vs B{1,2,3,4,5}, the bracketed expressions giving the order of play of each team. With reference to the skill matrix this particular scenario would result in a score of 5-0 in favor of team B. However, a slight change in team A to {5,1,2,3,4} would result in a score of 4-1 in favor of team A. It would seem, therefore, that the choice of fight order of the five players is of crucial importance in gaining the best possible final team score ... or is it? The first point to note is that in order to achieve the much improved score we have taken advantage of knowledge of the opposing teams fight order. This would not normally be available in a real tournament unless we are faced with a team that fields it’s players with predictable regularity.

Let us continue our analysis in more general terms by considering n-person teams rather than the specific case of five-person teams discussed above. The number of possible fight orders for each team is n! giving a total of (n!)² possible scenarios. Based on information from the skill matrix we can compute the outcome for each of these. Note that, in all cases, the sum of the two team scores is n, so the computation of outcomes provides us with a payoff matrix for a two-person constant-sum game.

The procedure to derive the payoff matrix from the skill matrix may be expressed formally as

\[ p_{ij} = \sum_{k=1}^{n} x_{k_i,k_j} \quad \text{for } i=1,\ldots,n, j=1,\ldots,n \]

where P is the payoff matrix for team A, X represents the skill matrix and A is a n! x n matrix containing a list of all possible fight orders. We are now in a position to invoke the standard LP formulation of a two-person constant-sum game as follows.
where \( v \) (unrestricted in sign) is the value of the game or, in this context, the optimum expected score for team A.

MATLAB code to compute the payoff matrix and solve the above model is here\(^{(1)}\). The formulation is rearranged into a minimization problem subject to less than or equal to constraints in order to comply with the input requirements of the MATLAB linear programming function (linprog.m) included in the Optimization Toolbox.

The following is an Excel spreadsheet layout that may be used to implement the above formulation for three-person teams.

![Excel Spreadsheet Layout](1)

This spreadsheet complete with solver instructions to generate the optimal solution is here\(^{(2)}\) and a rather unwieldy but workable five-person team version is here\(^{(3)}\).

It would appear that, based on a number of trials using randomly generated skill matrices, a random strategy with every fight order represented with equal probability is an optimal solution. This indeed turns out to be the case and a proof of a more general problem is given in Hurley (2002). He refers to this 'equal probability' solution as the pick-out-of-a-hat strategy.

This unexpected result suggests a useful teaching activity. Students may be presented with the scenario described above and encouraged to model and solve the problem. A number of skill matrices may be provided as test data to allow for recognition of the emergence of a 'general rule'. Finally, more mathematically able students may be encouraged to work towards a proof.

Although the 'equal probability' solution is optimal, it is not uniquely optimal. Quite frustratingly, the Excel models provided produce optimal solutions that do not include all strategies equally and hence would not facilitate recognition of the general rule discovered using the MATLAB model. Nevertheless, the writer chose to include this approach as it provides a challenging student exercise in spreadsheet modeling.

**Acknowledgement**

The writer would like to thank Ian McGowan of Lancashire Business School for a number of useful comments that were incorporated into the finished article.

**References**


\(^{(1)}\) See appendix.


\(^{(3)}\) [http://ite.pubs.informs.org/Vol5No1/Chlond/karate5.xls](http://ite.pubs.informs.org/Vol5No1/Chlond/karate5.xls)
Appendix

MATLAB code

function [X,value,R]=play_order(n,R)

% play_order computes optimum play order for n-person team
% Usage:
% [X,value,R]=play_order(n,R)
% Send:
% n - number of players in team
% R - skill matrix
% Returns:
% X - solution to zero-sum game
% value - value of game
% R - skill matrix

% M J Chlond - September 2004
% mchlond@lancs.ac.uk
% Lancaster Business School

% generate skills matrix unless provided
if nargin == 1
    R = round(rand(n)*2)/2
end

% some data preparation
nf = factorial(n);
P = zeros(nf);
A = perms(1:n);

% compute payoff matrix
for i=1:nf
    for j=1:nf
        for k=1:nf
            P(i,j)=P(i,j)+R(A(i,k),A(j,k));
        end
    end
end

% build LP
P = [-P ones(nf,1)];
b = zeros(nf,1);
f = [zeros(1,nf) -1];
Aeq = [ones(1,nf) 0];
beq = 1;
LB = [zeros(1,nf) -n];
UB = [ones(1,nf) n];

% solve LP
[X,value] = linprog(f,P,b,Aeq,beq,lb,ub);
X = X(find(beq==1));
value = -value;
return