QEL-Alogrithm

September 2021

1 Input generation

This section defines the basic input to the algorithm.

The input are three lists of data $C = P, M, S$:

- P_i labels the candidates of the physics specialty $i = 1...8$
- M_i labels the candidates of the management specialty $i = 9...16$
- S_i labels the candidates of software specialty $i = 17...20$

That is C_i runs over 20 entries: 1...8 for P, 9...16 for M and 17-20 for S. Each candidate entry $C = P, M, S$ consists of the following elements:

$$C_i = \{c_1, c_2, c_3, c_4 \; ; \; p_1, p_2, p_3\} \tag{1}$$

with $c_i = 1...5$ their preference for challenge i and $p_j = 1...20$ their three suggested partners (no preferred order).

Task: Use e.g. google collaboratory to open a python file and use " pd.read_excel(FILENAME)" to extract the $C = P, M, S$ lists from the google sheet. Then do a search to replace each partner ID with the corresponding p_i number between 1...20.

2 Preliminary visualisation

This section builds a way to visualise the interests of the candidates. This is not strictly needed to perform the algorithm. However, (a) it serves as a consistency check with the result of the algorithm later on and (b) often the human mind can work extraordinarily well with patterns and find good solutions on its own.

Each candidate entry gets enriched by the following elements:

$$C_i' := C_i \cup \{\vec{x}, d, \vec{x}'(i)\} \tag{2}$$

$$\vec{x} := \frac{1}{4}\begin{pmatrix} -(c_1-1)+(c_2-1)-(c_3-1)+(c_4-1) \\ (c_1-1)+(c_2-1)-(c_3-1)-(c_4-1) \end{pmatrix}, \tag{3}$$

$$d := 1 + |(x_1+x_2)(x_1-x_2)|/4, \tag{4}$$

$$\vec{x}'(i) := \vec{x}/d \tag{5}$$

Task: We set up a chart with range $[-1,1]^2$ and plot for each C_i its corresponding $\vec{x}'(i)$ with label P_i, M_i or S_i respectively.
For clarification it is probably useful to choose three different colours for P, M, S and to add to each $\vec{x}'$ a random vector with numbers in $\pm 0,05$ in order to avoid cluttering.

The plot showcases the affinities of the candidates for the different challenges, with the following legend:

- Left upper corner == challenge 1
- Right upper corner == challenge 2
- Left lower corner == challenge 3
- Right lower corner == challenge 4

Task: If the plot gets not too crowded (alternatively a second plot?) it may be useful to plot arrows between $\vec{x}'(i)$ for each candidate and the corresponding $\vec{x}'(p_i)$ of its partners p_i.

Task: Repeat everything but with

$$\vec{x} := \frac{1}{4}\begin{pmatrix} -(c_1-1)+(c_2-1)+(c_3-1)-(c_4-1) \\ (c_1-1)+(c_2-1)+(c_3-1)+(c_4-1) \end{pmatrix}, \tag{6}$$

$$\tag{7}$$

Comparison of both plots will resolve a certain degeneracy for undecided candidates.

3 Optimisation procedure

In this section, we define a *penalty function* for a given constellation and will then probe the parameter space to find the function that minimizes the penalty. The parameter space of all possible configurations is

$$4!\left(\binom{8}{2}\binom{6}{2}\binom{4}{2}\right)^2 = 1,5 * 10^8 \tag{8}$$

Thus it is clear that not all possible constellations can be checked and we can to probe only some of them.

Penalty function:
First, we extract the set of all pairs of candidates, that want to work together: Define an empty list $\mathcal{P} = \emptyset$. For each C_i iterate all p_i and check whether there exists $i \in \vec{p}$ of C_{p_i}. If yes, then add $k = (k_1, k_2) = (i, p_i)$ to $\mathcal{P}$.[1]
Second, we define a *constellation* as follows: a 20-dimensional vector $\vec{u}$, with $u_k \in \{1, 2, 3, 4\}$ indicating the challenge that is assigned to candidate k. Each constellation is subject to constraints: each number $\{1, 2, 3, 4\}$ appears (1) only once in $k = 17...20$, and (2) only twice in $k = 1...8$.
Now, we define the following penalty function:

$$pen_{tot}(\vec{u}) = pen(\{C_i\}) + pen'(\mathcal{P}) \tag{9}$$

$$pen(\{C_i\}) = \sum_{k \in 1...20} \left((5 - c_{u_k})^2 + \prod_{i=1,2,3} 2^{1-\delta(u_{p_i(k)}, u_k)} \right) \tag{10}$$

$$pen'(\mathcal{P}) = 9 \sum_{\vec{k} \in \mathcal{P}} (1 - \delta(u_{k_1}, u_{k_2})) \tag{11}$$

with $\delta(a, b)$ being the Kronecker-delta.

Prior: [If needed, we can add this at a later stage.]

Optimisation:
Given a constellation $\vec{u}$ we define the iterative optimisation run $\mathcal{L}$:
We start by defining $\vec{t} := \vec{u}$. Then we iterate the following loop:

- We built a list of 12 of new constellations $\vec{n}_k$:
  - For $k = 1...4$: choose randomly a pair of numbers between $i, j \in \{1...8\}$ then exchange $u_i \longleftrightarrow u_j$ and call this $\vec{n}_k$.
  - For $k = 1...4$: choose randomly a pair of numbers between $i, j \in \{9...16\}$ then exchange $u_i \longleftrightarrow u_j$ and call this $\vec{n}_{4+k}$.
  - For $k = 1...4$: choose randomly a pair of numbers between $i, j \in \{17...20\}$ then exchange $u_i \longleftrightarrow u_j$ and call this $\vec{n}_{16+k}$.
- We compute $mp := \text{Min}_{k \in 1...12} pen_{tot}(\vec{n}_k)$ and $cp := pen_{tot}(\vec{t})$.
- We make the decision on whether the run finishes
  - If $mp \leq cp$ then

$$\vec{t} := \vec{n}_k \text{ such that } pen_{tot}(\vec{n})_k = lm \tag{12}$$

    and we repeat the loop

[1] Currently, this implies that each pair appears twice. This considered in the penalty function below.

– Else $\mathcal{L}(\vec{u}) := \vec{t}$ and the loop terminates.

Task: We choose randomly N-many (e.g. 10) constellations $\vec{u}(j)$ with $j = 1...N$, each constellation subject to the above-mentioned constraints (and the priors). Then, we compute $\mathcal{L}(\vec{u}(j))$ for each j. The final output is

$$\vec{u}_{final} := \mathrm{Min}_j \;\; pen_{tot}(\mathcal{L}(\vec{u}(j))) \tag{13}$$