Intelligence Techniques Are Needed to Further Enhance the Advantage of Groups with Diversity in Problem Solving

Oscar Castillo and Patricia Melin
Tijuana Institute of Technology, Mexico

J. Esteban Gamez, Vladik Kreinovich, and Olga Kosheleva
University of Texas at El Paso, USA

1. Introduction to the Problem

- *Empirical fact:* diversity in a group often enhances the group's ability to solve problems.
- *Theoretical explanation* (S. E. Page): diverse groups *can* outperform groups of high-ability problem solvers.
- *Problem:* algorithmic diversity rules (like quotas) are not always successful.
- *Our approach:* we consider the problem of designing the most efficient group as an optimization problem.
- *Our result:* this optimization problem is computationally difficult (NP-hard).
- *Conclusion:* it is not possible to come up with simple algorithmic rules for designing such groups.
- *Conclusion:* we must combine standard optimization techniques with expert knowledge.

2. Towards the Formulation of the Problem in Exact Terms

- From n individuals $\{1, \ldots, n\}$, we must select the most efficient group $G \subseteq \{1, \ldots, n\}$ for solving the problem.
- For each i, we set $x_i = 1$ is the i-th person is selected, and $x_i = 0$ otherwise.
- For simple mechanical work, group efficiency is the sum of productivities: $p = \sum\limits_{i \in G} p_i = \sum\limits_{i=1}^{n} p_i \cdot x_i$.
- For more complex tasks, interaction can either help ($p_{ij} > 0$) or inhibit efficiency ($p_{ij} < 0$).
- After the linear approximation, the next approximation is quadratic:

$$p = \sum_{i=1}^{n} p_i \cdot x_i + \sum_{i \neq j} p_{ij} \cdot x_i \cdot x_j.$$

3. Explanations: Why Help and Why Inhibition

- *Homogeneous group:* individuals with similar ways of thinking and with similar skills.
- *Property:* there is not much that these individuals can learn from each other.
- *Simple case:* the problem is easy to subdivide into sub-problems.
- *Typical case:* the problem is not easy to subdivide.
- *Result:* the solvers follows similar paths, duplicate work.
- *Productivity:* same as for one solver: $p \approx p_i < p_i + p_j$, i.e., $p_{ij} < 0$.
- *Diverse group:* individuals complement each other, learn from each other.
- *Result:* productivity increases: $p > p_i + p_j$, i.e., $p_{ij} > 0$.

4. Problem of Selecting the Most Efficient Group: Precise Formulation

- *Given:*
  - an integer $n > 0$;
  - rational numbers $p_1, \ldots, p_n$, and
  - rational numbers r_{ij}, $1 \le i, j \le n$, $i \ne j$.
- *Find:* the combination of n values $x_1 \in \{0, 1\}$, $\ldots$, $x_n \in \{0, 1\}$ for which the expression

$$p = \sum_{i=1}^{n} p_i \cdot x_i + \sum_{i \ne j} p_{ij} \cdot x_i \cdot x_j$$

  is the largest possible.
- *Our result:* the problem of selecting the most efficient group is NP-hard.

5. Maybe a Less Ambitious Problem Will Be Easier to Solve?

- *We wanted:* the most efficient group, with the largest possible productivity.
- *Problem:* the related task is computationally difficult (NP-hard).
- *Natural idea:* maybe a less ambitious problem will be easier to solve.
- *Specific suggestion:*
  - instead of looking for a group withe the largest possible productivity,
  - look for a group with a desired level of productivity $p \geq p_0$.
- *Problem:* as we will see, this "relaxed" problem is still computationally difficult (NP-hard).

6. Problem of Selecting a Group with a Given Efficiency

- *Given:*
  - an integer $n > 0$;
  - rational numbers $p_1, \ldots, p_n$,
  - rational numbers r_{ij}, $1 \le i, j \le n$, $i \ne j$, and
  - a rational value p_0.
- *Find:* the combination of n values $x_1 \in \{0,1\}, \ldots, x_n \in \{0,1\}$ for which

$$p \stackrel{\text{def}}{=} \sum_{i=1}^{n} p_i \cdot x_i + \sum_{i \ne j} p_{ij} \cdot x_i \cdot x_j \ge p_0.$$

- *Our result:* The problem of selecting a group with a given efficiency is NP-hard.

7. A Simple Illustrative Example

- *Description:* we have two groups of equally productive people, $G_1 = \{1, \ldots, m\}$ and $G_2 = \{m+1, \ldots, 2m\}$:
$$p_1 = \ldots = p_m = p_{m+1} = \ldots = p_{2m} = 1.$$
- *Help:* persons from the same group collaborate: $p_{ij} = 1$ for $i, j \le m$ or $i, j > m$.
- *Inhibition:* between-group tension decreases productivity: $p_{ij} = p_{ji} = -a$ for $i \le m$ and $j > m$.
- *We select:* m_1 folks from G_1 and m_2 folks from G_2.
- *Resulting productivity:* $p = m_1^2 + m_2^2 - a \cdot m_1 \cdot m_2$, where $m_i \in [0, m]$.
- *Solution* (Case $a < 1$): the most diverse group $m_1 = m_2 = m$ is the most efficient.
- *Comment:* when $a > 1$, tensions are so high that homogeneous groups are better: e.g., $m_1 = m$ and $m_2 = 0$.

8. Conclusions

- One of the objectives of fuzzy techniques: to formalize the meaning of words from natural language.
- Examples: "efficient", "diverse", etc.
- The main use of fuzzy techniques: formalize expert knowledge expressed in natural language.
- In this paper, we have shown that
  - if we do not use this knowledge, i.e., if we only use the data,
  - then selecting the most efficient group is computationally difficult (NP-hard).
- Thus, the need to select efficient groups in reasonable time justifies the use of fuzzy (intelligent) techniques.
- Moreover, there is a need to combine intelligent techniques with more traditional optimization techniques.

9. Acknowledgments

This work was supported in part:

- by NSF grant HRD-0734825 and
- by Grant 1 T36 GM078000-01 from the National Institutes of Health.

10. When is an Algorithm Feasible?

- The notion of NP-hardness is related to the known fact that some algorithms are feasible and some are not.
- Whether an algorithm is feasible or not depends on how many computational steps it needs.
- *Case 1:* for some input x of length $\text{len}(x) = n$, an algorithm requires 2^n computational steps.
- *Example:* for an input of a reasonable length $n \approx 300$, we need 2^{300} computational steps.
- *Problem:* this takes longer than the Universe's lifetime.
- *Conclusion:* this algorithm is not feasible.
- *Case 2:* an algorithm requiring n^2 or n^3 steps is usually feasible.
- *Resulting definition:* an algorithm is feasible if its running time $t(n)$ is bounded by a polynomial $P(n)$.

11. NP: Class of General Problems

- *General formulation:*
  - *we have* some information x;
  - *we need to find* y which satisfies the feasible-to-check property $R(x, y)$.
- *Example from mathematics:*
  - *given:* a mathematical statement x;
  - *find:* a proof y of x or of "not x".
- *Comment:* computers can easily check step-by-step proofs y, but finding a proof is a challenge.
- *Engineering:* find a design y that satisfies given specifications x.
- *Physics:* find a formula y that is consistent with all the observations x.

12. Class NP and the Notion of NP-Hardness

- Every problem from the class NP can be solved by exhaustively checking all 2^n possible solutions y.
- *Example:* try all possible combinations of n symbols until we find a proof.
- *Open question:* it is not known whether a feasible algorithm can solve all NP problems.
- *What is known:* some NP problems are more difficult than others ("NP-hard").
- *Precise meaning:* every problem from NP can be reduced to this problem.
- *How to prove NP-hardness:* reduce one of the known NP-hard problems $\mathcal{P}_k$ to the desired one $\mathcal{P}_d$.
- *Proof:* every $\mathcal{P} \in \mathrm{NP}$ can be reduced to $\mathcal{P}_k$, and $\mathcal{P}_k$ can be reduced to $\mathcal{P}_d$, so $\mathcal{P}$ can be reduced to $\mathcal{P}_d$.

13. Proof of Our Result: Main Ideas

- We prove NP-hardness of our problem by reducing the following known NP-hard problem to it.
- The *subset sum* problem:
  - *given:* n positive integers $s_1, \ldots, s_n$;
  - *find:* the signs $\varepsilon_i \in \{-1, 1\}$ for which $\sum\limits_{i=1}^{n} \varepsilon_i \cdot s_i = 0$.
- *Reduction* means that:
  - to every instance $s_1, \ldots, s_n$ of the subset sum problem,
  - we must assign (in a feasible, i.e., polynomial-time way) an instance p_0, p_i, p_{ij} of our problem,
  - in such a way that the solution to the new instance will lead to the solution of the original instance.

14. Proof (cont-d)

- *Original problem:* find $\varepsilon_i \in \{-1, 1\}$ s.t. $\sum\limits_{i=1}^{n} \varepsilon_i \cdot s_i = 0$.
- *New problem:* find $x_i \in \{0, 1\}$ for which

$$p \stackrel{\text{def}}{=} \sum_{i=1}^{n} p_i \cdot x_i + \sum_{i \neq j} p_{ij} \cdot x_i \cdot x_j \geq p_0.$$

- A solution $x_i \in \{0, 1\}$ of the new instance must lead to the solution $\varepsilon_i \in \{-1, 1\}$ of the original instance.
- *Natural idea:* take $\varepsilon_i = 2 \cdot x_i - 1$.
- *Natural reduction:* take $p = p_0 - \left(\sum\limits_{i=1}^{n} \varepsilon_i \cdot s_i\right)^2$.
- *Why it works:* $p \geq p_0 \Leftrightarrow \sum\limits_{i=1}^{n} \varepsilon_i \cdot s_i = 0$.
- *Specifics:* $p_0 = \sum\limits_{i=1}^{n} s_i$, $p_i = 4 \cdot s_0 \cdot s_i - 4 \cdot s_i^2$, $p_{ij} = -4 \cdot s_i \cdot s_j$.

15. Discussion

- Strictly speaking, we have proved NP-hardness of a specific choice of the quadratic function $p(x_1, \ldots, x_n)$.
- However, one can easily check that
  - if a problem $\mathcal{P}_0$ is NP-hard,
  - then a more general problem $\mathcal{P}_1$ is NP-hard as well.
- Thus, we have indeed proved that the (more general) problem is also NP-hard.