Leveraging Tradespace-Exploration for A Senior Project Team Formation Application

By: Miguel Saenz, Ged Mulingtapang, Professor Hyeonik Song

Introduction

- Problem Statement
  - The goal of this project was to construct and refine an application that leverages a genetic algorithm to perform tradespace exploration and automatically execute team formation for the ME senior design project classes, ME 428, ME 429, and ME 430
- Prior Work/Literature Review
  - Paper ***[1]*** provided an understanding of VASSAR, the rule-based system that we leveraged for this app
  - Papers ***[2]***, ***[3]***, and ***[4]*** provided greater clarity on the more concrete goals and metrics for said goals (see below)

Objectives

- Abstract Goals
  - Make app optimally functional and performant
  - Expand Pareto Frontier to give users of the app a number of optimal solutions to choose from.
  - Give preference to the formation of less performant but more feasible groups in our calculations.
- Concrete Goals
  - Make app functional for all three relevant datasets provided
  - (One test rulesheet/dataset and two real rulesheets/datasets)
  - Make app run in a reasonable timespan (≤ 10 minutes for worst-case dataset and n*log(n*m) in the average or worst case generally.
  - Make optimal parameter allocation automatic in the application so that users need not make any modifications for a reasonably performant solution unless they explicitly elect to.

Methodology

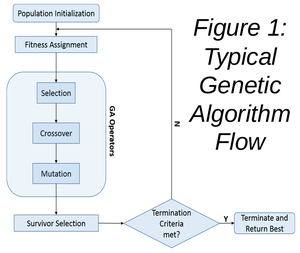


Figure 1: Typical Genetic Algorithm Flow

- What we focused on improving:
  - Population Initialization
    - What is the ideal population and generation size?
  - Fitness Assignment
    - How should we score teams, assign penalties, skew fitness for feasibility, etc.?

Figure 2: Typical App Run-through

Acknowledgement:

This research was funded by Julie Ma, Time Bienz, and the College of Engineering

Methodology (Continued)

Population/Generation Size

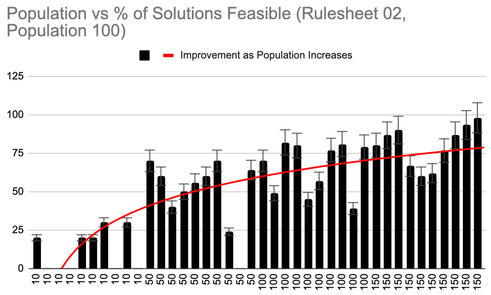


Figure 3: Population vs Feasibility

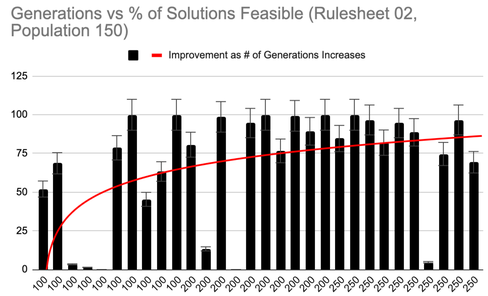


Figure 4: Generations vs Feasibility

- **Takeaways** (from these and other datasets not pictured for lack of space)
  - Both higher population and generation values improve frequency of feasible solutions and consistency of creation of feasible solutions.
  - Both of these values have a diminishing return as one increases the value, hence there must be a logarithmic functionality we can use to improve consistency and frequency of feasible solutions without exponentially increasing runtime (each additional member of the population persists across generations, and hence increases computational time for eachgeneration, and vice versa).
- **Improvements**
  - Automatically calculate ideal population, generation and penalty #s with below formulas:
    - **Generation: 20 * log(# of students) * (# of projects)**
    - **Population: 125 * log(# of students) * log(# of projects)**
    - **Penalty Weight: 14 * [(# of students) - (# of projects)]**
  - All solutions use a constant to ensure a reasonable minimum as well as take inspiration from the equation included in ***[2]*** **[# of generations = K * (# of individuals) log(# of groups)]** that leverages logarithmic scaling to flatten runtime as the number of groups increases.

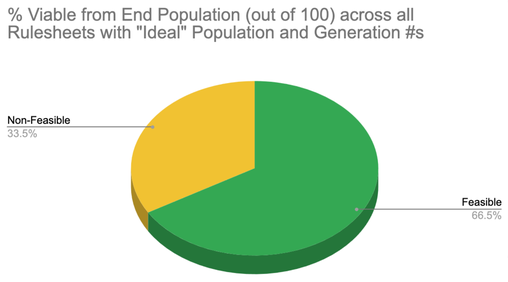


Figure 5: Percent feasible after automatic calculation improvements across all datasets

- **Takeaways**
  - This data is deceptively positive, as while on average across rulesheets we are providing solutions where nearly 3/4s of our population is viable, the larger real dataset (rulesheet_05) no feasible solutions were found across all runs.
  - The observed problem was with large datasets larger groups with a highly performant project preference and membership score would overwhelm the penalty that was calculated for the number of groups that didn't contain the ideal # of students
- **Improvements**
  - Created a cubic scaling penalty based on how far away from the ideal size a group is to push solutions much harder in the direction of feasability over raw performance as far as member and project preference.
  - penalty = ((# in group – ideal)^3 + 3 * penalty weight)

User Interface

- Purpose:
  - The goal of this project's UI design was to help our targeted audience of engineering professors at Cal Poly efficiently navigate this application to execute team formation for the senior design project class
- Three Key Elements
  - Organization
  - Ease of use
  - Engagement

Developments

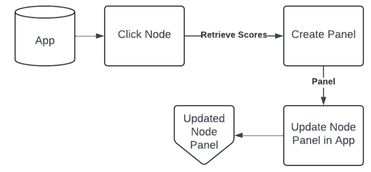
- Three Main Features
  - Ranking table
  - Node clicking panel (not yet developed)
  - Save solutions database (not yet developed)

Rules | Optimizer | Ranking Table

Project Preference Score	Member Preference Score
53	60
52	60
54	60
52	60
53	60
53	60
54	60
53	60
55	60
55	60

Figure 5: table that ranks the best solutions from highest member preference score to lowest

Figure 7: diagram for a node panel feature that shows a solution's member preference score and project preference score when the node is selected



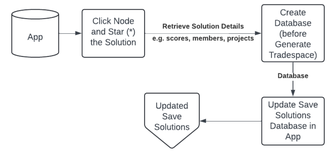


Figure 8: diagram for a database feature that stores solutions that are specifically favorited by the user

Conclusions

- **Final State of App**
  - Leveraging all the aforementioned improvements we were able to attain about an average of 35% of solutions being feasible in Rulesheet_05 runs. This boosted our overall feasibility across all datasets to nearly 80%.
  - Overall, the consistency with which we are able to produce feasible solutions that users can then modify with the evaluate functionality to better fit their use case is more than acceptable, and the max measured runtime with the largest real dataset we tested upon (rulesheet_05) standing at 7 minute is also more than acceptable for a program professors will likely run no more than a handful of times a year.

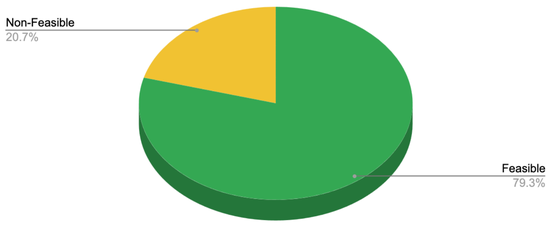


Figure 6: Percent feasible after penalty calculation improvements across all datasets

References

Images:

Figure 2: Deshpande, A., & Kumar, M. (2018). *Artificial Intelligence for Big Data: Complete guide to automating big data solutions using artificial intelligence techniques*. Packt Publishing.

Papers:

[1]D. Selva and E. F. Crawley, "VASSAR: Value assessment of system architectures using rules," 2013 IEEE Aerospace Conference, Big Sky, MT, USA, 2013, pp. 1-21, doi: 10.1109/AERO.2013.6496936.

[2] Esgario, J. G. M., E, D. S. I., & Krohling, R. A. (2019, March 8). Application of genetic algorithms to the multiple team formation problem. arXiv.org. https://arxiv.org/abs/1903.03523

[3] A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics, Computers & Education, Volume 58, Issue 1, 2012, Pages 560-569, ISSN 0360-1315, https://doi.org/10.1016/j.compedu.2011.09.011. (https://www.sciencedirect.com/science/article/pii/S0360131511002284)

[4] H. Wang, J. Li, Y. Song, J. Huang, J. Li and Y. Chen, "An Improved Genetic Algorithm for Team Formation Problem," 2022 IEEE Symposium Series on Computational Intelligence (SSCI), Singapore, Singapore, 2022, pp. 774-781, doi: 10.1109/SSCI51031.2022.10022143.