Constraint Handling in Genotype to Phenotype Mapping and Genetic Operators for Project Staffing

Soo Ling Lim
Department of Computer Science
University College London
United Kingdom
s.lim@cs.ucl.ac.uk

Yi Kuo
Department of Computer Science
University College London
United Kingdom
ucabyku@ucl.ac.uk

Peter J. Bentley
Department of Computer Science
University College London
United Kingdom
p.bentley@cs.ucl.ac.uk

ABSTRACT

Project staffing in many organisations involves the assignment of people to multiple projects while satisfying multiple constraints. The use of a genetic algorithm with constraint handling performed during a genotype to phenotype mapping process provides a new approach. Experiments show promise for this technique.

CCS CONCEPTS

• **Computing methodologies** → Genetic algorithms; • **Information systems** → Enterprise resource planning; • **Social and professional topics** → Project staffing

KEYWORDS

Genetic algorithms, constraint handling, human resource allocation

ACM Reference format:

S. L. Lim, Y. Kuo, and P. Bentley. 2020. Constraint Handling in Genotype to Phenotype Mapping and Genetic Operators for Project Staffing. In *Proceedings of ACM GECCO conference, Cancun, Mexico, July 2020 (GECCO'20 Companion)*, 2 pages.
DOI: 10.1145/3377929.3398165

1 INTRODUCTION

The need to assign people to projects is a common problem [1, 2]. For example, a consulting company may have 5 upcoming projects and 40 available consultants, with each project requiring a specific number of consultants. During such assignment, many constraints must be satisfied, such as the consultant's availability, expertise fit for the project, skills to improve and develop, preferences (e.g., travel, interests), while considering their past performance, compatibility and the project difficulty. Our work tackles the problem of optimisation of human resource allocation within an organisation (project staffing), taking such constraints into account.

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3398165

We define the general problem as follows: k sets each of n_i people where i denotes set number and $0 < i <= k$, are picked from a central pool of m people and added to k existing teams each of t_i people, where $n_i >= 1$, $t_i >= 0$, $m >= \sum_{i=0}^{k} n_i$, simultaneously optimising for criteria such as availability, experience, skills, and predicted team performance based on individual personality [3], where the person staffing the project could choose which criteria are more important, have clear scores for each criteria for any team and satisfying multiple constraints.

For this work, we focus on the following constraints. Each project must have the correct number of people assigned to it, this number may be different per project. A person cannot be assigned to more than one project. A person must provide a different preference per project. People should be assigned to projects with a bias towards their performance – if two or more people prefer the same project, the person with better performance has priority.

Traditionally constraint satisfaction within genetic algorithms may be performed by using smart genetic operators (for hard constraints) and fitness penalties (for soft constraints) [4]. Here we explore the use of constraint handling during the genotype to phenotype mapping process, supported by smart operators. We aim to constrain the search less (and potentially enhance search through the creation of neutral networks [5]), and ensure that all genotypes are mapped to valid phenotypes that satisfy the constraints.

2 METHOD

2.1 System

A standard canonical genetic algorithm is used to optimise the teams by picking an optimal combination of people for each team. In order to enable constraint handling, a novel data representation was designed to represent candidate the project preferences for candidate team members, with modified genetic operators.

In our work we model a consultancy (or a technology company running a large number of agile teams) where each n_i may range from 5 to 7 people, m may range from 40 to 100 people, each t_i is 0 and $k=5$ projects were used. The teams were optimised using metrics for each person and for each project: availability, industry fit, legal requirements, travel, project type fit, skill development, language, is difficulty appropriate, for each person. A dataset with plausible scores within the ranges provided above was created. The following sections describe the details of the algorithm.

2.2 Representation

The genotype representation is shown in Table 1. Random initialisation is used, with constraint handling, i.e., for every person to be assigned, every project is given a unique, non-repeated, random preference score.

Table 1: Chromosome representation for each individual comprising the preference for each project number.

Person	Preference per project				
	1	**2**	**3**	**4**	**5**
0	3	1	2	5	4
1	2	5	3	3	1
2	1	3	5	2	4
…	…	…	…	…	…
m	5	4	1	3	2

2.3 Mapping with Constraint Handling

In order to evaluate the quality of each solution, genotypes must be mapped to phenotypes. This is performed as follows.

People are assigned to each project by their preference. For each project, for each preference value, add every person *n* who listed the project as their preference *pf* (where lower *pf* indicates higher preference), sorted by decreasing performance, e.g.,

$Project_1$ = {n_2[*pf*=1], n_1[*pf*=2], n_0[*pf*=3], …}

$Project_2$ = {n_0[*pf*=1], n_2[*pf*=3], n_1[*pf*=5], …}

…

$Project_5$ = {n_1[*pf*=1], n_0[*pf*=4], n_2[*pf*=4], …}

For each project, if |Project| = required size *S* then that project is successfully allocated. However, if |Project| >= required size *S* then only the first *S* people are allocated. Any projects containing fewer than *S* people are then filled using unallocated people according to their *pf* values and performance.

2.4 Evaluation and Selection

The phenotypes are evaluated by summing scores for 8 metrics since all phenotypes with higher values are better. They are: (1) availability [0–1]: calculated based on the individual's available date and the project start date, (2) industry fit [0–1]: calculated based on the number of past projects that the individual has done that is in the same industry as the current projects, (3) legal requirements [0 or 1]: whether the project's legal requirements match the individual, (4) travel [0 or 1]: if the project requires travelling and the individual does not want to travel, 0 else 1, (5) project type fit [0–1]: calculated based on the number of past projects that the individual has done that is in the same type as the current projects, (6) skill development [0 or 1]: whether the project improves the skills that the individual would need developed, (7) language [0 or 1]: if the individual is fluent in the language required by the project, 1 else 0, and (8) is difficulty appropriate [0 or 1]: calculated based on individual's position, difficulty of recent projects, and difficulty of current project.

Once fitness values have been assigned to each phenotype, the population is sorted in the order of fitness and the fittest $N_{parents}$ parents are chosen. Random pairs are chosen, with one offspring created at a time using crossover and mutation operators, until a new population of solutions has been generated.

2.5 Genetic Operators (Mutation and Crossover)

Given two parents A and B, in order to make child C using crossover, for every person in the chromosome, pick a random project choice for child C, make it equal to either the project choice from A or B with a probability of 0.5. If this choice has already been taken in any other project choice of C, then the other parent's project choice is used. If this choice has also already been taken by C, then we take a random choice not already taken. Repeat until all project choices for C has been taken for that person, and repeat for all people in the chromosome.

For mutation, a child solution is chosen with probability of 0.8. Within the genotype, a random person is chosen, two random project choices are chosen, and swapped.

3 DISCUSSION

The system was run on random data and on the hand-designed data described in Section 2.1. Optimisation achieved excellent results, with better results evident for the non-random data (as random data is more likely to have conflicting constraints), see Figure 1.

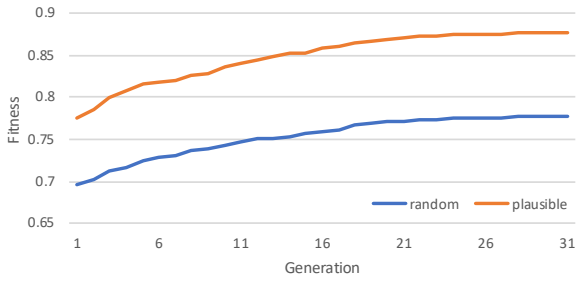

Figure 1: Representative runs showing fitness per generation for the two datasets.

This work is ongoing. Future work will add more constraints to the team assignment problem, for example, pairs of people must be kept together (or kept apart), people may work part time on multiple projects, people with specific roles must be included.

REFERENCES

[1] C. Heimerl and R. Kolisch, 2010. Scheduling and staffing multiple projects with a multi-skilled workforce. OR Spectrum, 32(2), pp. 343-368.

[2] A. Barreto, M. O. Barros, and C. M. L. Werner, 2008. Staffing a software project: a constraint satisfaction and optimization-based approach. Computers & Operations Research, 35(10), pp. 3073-3089.

[3] S. L. Lim and P. J. Bentley, 2018. Coping with uncertainty: modelling personality when collaborating on noisy problems. In Proceedings of the 2018 Conference on Artificial Life (ALIFE), Tokyo, Japan, pp. 566-573.

[4] T. Yu and P. J. Bentley, 1998. Methods to evolve legal phenotypes. In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN), Amsterdam, pp. 280-282.

[5] M. Ebner, M. Shackleton and R. Shipman, 2001. How neutral networks influence evolvability. Complexity, 7(2), pp. 19-33.