GRASP - based Mentorship and Teamwork Problem (Google HashCode 2022)

Egzona Vllasaliu[1[220756200017]]

[1] Faculty of Electrical and Computer Engineering, University of Prishtina,
Str. "Agim Ramadani", p.n. 10 000 Prishtinë, Republika e Kosovës

Keywords: Project Assignment · Greedy Randomized Adaptive Search Procedure · Teamwork · Mentorship

1 Introduction and problem description

Project Assignment in the field of software development has been a topic of interest with varying parameters, constraints, and objectives. This paper presents a solution approach for a variant of the problem wherein each project has its own set of required skills, preferred completion time, and a scoring mechanism. Software developers, each with their unique set of skills and experience levels, are to be assigned to these projects to maximize the overall project scores.

This problem has elements of graph coloring where the developers are nodes, and edges are formed between developers having at least one skill in common. The problem also shares attributes of the packing problem as we aim to pack the most suited developers into each project, considering each project's unique requirements.

Metaheuristics are powerful tools for solving complex optimization problems that are difficult to solve by traditional optimization methods. They are iterative search processes that efficiently explore the solution space to find near-optimal solutions (Glover & Kochenberger, n.d.). Examples include but are not limited to evolutionary algorithms, swarm optimization, and tabu search. This document delves into the application of GRASP, a metaheuristic known for its efficiency and simplicity.

GRASP, or Greedy Randomized Adaptive Search Procedure, is a metaheuristic algorithm used for solving discrete optimization problems. It consists of two phases, a construction phase, where a feasible solution is iteratively constructed, and a local search phase, where the neighboring solution space of the constructed solution is searched for improvements (Luke, 2014). The effectiveness of GRASP comes from the interplay between these phases, allowing it to balance exploration and exploitation in the solution space.

2 Solution method

GRASP has been successfully applied in a wide variety of optimization problems. From scheduling problems, routing problems to resource allocation problems, the simplicity and flexibility of GRASP make it a popular choice for solving complex problems. Specific case studies and problem-specific adaptations of GRASP will be discussed in this section.

The solution approach utilizes the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm. In our implementation, GRASP iteratively constructs solutions by making probabilistic decisions at each step. It utilizes a multi-neighbourhood approach to explore various solutions in the search space and picks the best among them, thereby leading to a high-quality solution over time. Our implementation also employs a memoization technique to cache and quickly retrieve the best developers for each project.

The solution also encourages a sense of teamwork and mentorship among developers. As they are assigned projects based on their skills, developers have the opportunity to work together, learn from each other, and improve their skills over time. This strategy not only provides an efficient solution but also creates an environment conducive to growth and learning for developers.

```
C ← {C1, ..., Cn} components
p ← percentage of components to include each iteration
m ← length of time to do hill-climbing

Best ← □
repeat
    S ← {}                                                   ▷ Our candidate solution
    repeat
        C' ← components in C − S which could be added to S without being infeasible
        if C' is empty then
            S ← {}                                           ▷ Try again
        else
            C'' ← the p% highest value (or lowest cost) components in C'
            S ← S ∪ {component chosen uniformly at random from C''}
    until S is a complete solution
    for m times do
        R ← Tweak(Copy(S))    ▷ Tweak must be closed, that is, it must create feasible solutions
        if Quality(R) > Quality(S) then
            S ← R
    if Best = □ or Quality(S) > Quality(Best) then
        Best ← S
until Best is the ideal solution or we have run out of time
return Best
```

3 Preliminary experimental results

Preliminary results demonstrate the efficiency of our approach, producing high-quality solutions for the Project Assignment problem. Our algorithm is compared with previous strategies based on various parameters, including total score, total project completion time, and developer skill improvement. Detailed results will be presented at the conference.

Table 2. Preliminary results. Best available solutions are from:
https://codingcompetitions.withgoogle.com/hashcode/round/00000000008caae7

Groups	A - an example	B - Better start small	C - Collaboration	D - Dense schedule	E - Exceptional skills	F - Find great mentors	Score
Make love, not war	33	969087	229517	674945	1640454	706200	**4220236**
Rethinkers	33	932759	274679	384328	1599952	904867	**4096618**
Past glory	33	909802	223267	562814	1640172	756170	**4092258**
Proof by submission	33	800991	259630	173626	1587033	1194515	**4015828**
Code	33	900799	259165	399105	1640492	765794	**3965388**
Us	33	885736	6184	91111	1255687	226272	**2069023**

4 Future work

This work is a stepping stone towards more comprehensive solutions. We aim to enhance the neighborhood relations within our GRASP algorithm, develop variants and hybrid metaheuristics, and perform an extensive analysis of the instance space for algorithm selection. We are also keen on investigating further improvements on the teamwork and mentorship aspect within the project assignments.

In detail, we plan on implementing a chain-like mechanism for developer assignment that allows for skill improvements and conflict resolutions. Developers with specific skills would be encouraged to mentor other team members, improving the overall team competency over time.

The GRASP methodology, while being robust and effective, still has ample scope for enhancements and adaptations. As we look forward, several areas of research and application emerge as promising avenues for the application and development of GRASP.

a. **Hybrid GRASP Algorithms**
One area of potential exploration is the combination of GRASP with other metaheuristics to form hybrid algorithms. While GRASP has been successful in solving numerous problems, integrating it with other metaheuristics like genetic algorithms or particle swarm optimization may result in algorithms capable of exploring the solution space more efficiently or finding higher quality solutions.

b. **Parallel GRASP**
With the increasing availability and affordability of high-performance computing resources, parallel versions of GRASP present an interesting avenue for future work. Parallel versions could potentially speed up the search process significantly and tackle larger, more complex problems.

c. **Adaptive GRASP**
Adaptive methods to adjust the parameters of GRASP during the execution of the algorithm could be another significant area of future work. Techniques that allow the algorithm to learn from its past performance and adapt its parameters could increase its efficiency and effectiveness.

d. **Real-world Applications**
Future work could also focus on applying GRASP to new types of real-world problems. Given the flexibility and simplicity of the method, it can potentially be adapted to solve emerging optimization problems in areas such as logistics, machine learning, energy systems, and more.

e. **Theoretical Analysis**
While a lot of work has been done on the practical application of GRASP, there is still scope for more theoretical work on understanding the algorithm's behavior, performance guarantees, and conditions under which it performs best.

References

1. F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," Control and Cybernetics, vol. 39, no. 3, pp. 653–684, 2000.
2. R. Martí, M. Resende, and C. Ribeiro, "Multi-start methods for combinatorial optimization," European Journal of Operational Research, vol. 226, no. 1, pp. 1–8, 2013.
3. Luke, S. (2014). Essentials of Metaheuristics. (2nd ed., Online Version 2.1). George Mason University. Department of Computer Science.
4. Glover, F., & Kochenberger, G. A. (Eds.). (n.d.). Handbook of Metaheuristics. Kluwer Academic Publishers.