Master Project Business Analytics

Optimizing the consultant-to-project assignment problem using Hybrid-Evolutionary Algorithms

Written by:
Sander Steur
Studentnr. 2675178

Supervisors:	Karine Miras & Guszti Eiben	
Second reader:	Alessandro Zocca	
Daily supervisor 1:	Joost van der Ploeg	(BearingPoint)
Daily supervisor 2:	Bram Zentveld	(BearingPoint)

A thesis submitted in fulfillment of the requirements for Master of Science degree in Business Analytics - Optimization of Business Processes

November 28, 2024

Abstract

This research addresses the Consultant-to-Project Assignment (C2Pa) problem at BearingPoint Netherlands, a challenge in the consultancy industry that focuses on optimizing consultant assignments to various projects to enhance operational efficiency. The traditional approach relies heavily on the intuition and experience of the Operational Team Lead (OTL) Team, making it a time-consuming and potentially biased process. This study investigates the implementation of meta-heuristic techniques to solve the C2Pa problem, specifically examining the efficiency and efficacy of local search.

Key Performance Indicators including Consultant Satisfaction, Skill Match, Hourly Cost, Utilization Rate, and the number of declined projects were used to evaluate assignment effectiveness. A meta-heuristic framework for simultaneous scheduling and staffing problems was developed, building upon random key representation that incorporates project selection, consultant assignment, skill division, and project scheduling. Two meta-heuristic algorithms were implemented: a Biased Random Key Genetic Algorithm (BRKGA) and a Scatter Search Algorithm (SS).

Experimental results demonstrate that the Scatter Search models outperform the BRKGA models in their current implementation. The implementation of local search techniques did not yield statistically significant improvements and significantly reduced efficiency. The model has proven its scalability by effectively handling large problem instances. The study recommends utilizing a fixed 2-week start time window for project scheduling to enhance consultant satisfaction and skill matching while maintaining cost-effectiveness. Additionally, implementing business rules that restrict the allowed service line and introduce minor flexibility in job positions of a team member role led to a robust model that aligns with the recommendations of the OTL team.

Contents

List of Figures **iv**

List of Tables **vi**

1 Introduction **1**

1.1 Problem Description 1

1.2 Relevance of the problem 5

1.3 Research Questions 6

1.4 Research Outline 6

2 Problem Context **8**

3 Literature Review **11**

3.1 Problem base 11

3.2 Application of Meta-heuristics 13

3.2.1 RCPSP 13

3.2.2 RCMPSP 15

3.2.3 MSRCPSP 16

3.2.4 MSRCMPSP 18

3.3 Conclusion 19

4 Data **21**

4.1 Consultant Capability Data 21

4.2 Consultant Availability Data 22

4.3 Project Data 23

5 Existing MILP model **27**

6 Methodology 30
6.1 Encoding . . . 30
6.1.1 Search Space Reduction . . . 31
6.1.2 Project Selection . . . 31
6.1.3 Consultant Selection . . . 32
6.1.4 Skill Division . . . 33
6.1.5 Time Windows . . . 33
6.2 Decoding . . . 34
6.3 Objective Function . . . 37
6.3.1 Satisfaction KPI . . . 37
6.3.2 Skill Match KPI . . . 38
6.3.3 Hourly Cost KPI . . . 40
6.3.4 Utilization Rate KPI . . . 40
6.3.5 Weighting the KPIs . . . 41
6.3.6 Fitness Function . . . 43
6.4 Local Search . . . 44
6.4.1 Consultant Swap . . . 45
6.4.2 Skill Swap . . . 46
6.5 BRKGA . . . 49
6.5.1 Methodology . . . 49
6.5.2 Hyperparameter Tuning . . . 53
6.6 Scatter Search . . . 56
6.6.1 Methodology . . . 56
6.6.2 Hyperparameter Tuning . . . 60

7 Evaluation of Results 62
7.1 Verification of Models . . . 63
7.1.1 Skill Match KPI . . . 63
7.1.2 Satisfaction KPI . . . 64
7.1.3 Hourly Cost KPI . . . 65
7.1.4 Utilization KPI . . . 66
7.2 Validation of Models . . . 68
7.3 Comparison to MILP . . . 69
7.4 Experiment Instance Size . . . 70
7.4.1 Instance size 3 . . . 72

7.4.2 Instance size 6 75
7.4.3 Instance size 12 78
7.4.4 Instance size 24 81
7.4.5 Conclusion Instance Size Experiment 84
7.4.6 Analysis Convergence Speed 84
7.5 Experiment Flexible Time Windows 85
7.6 Experiment Business Rules 88
7.6.1 STRICT Rules 89
7.6.2 SL+POS Rules 89
7.6.3 SL Rules 90
7.6.4 POS Rules 91
7.6.5 No Rules 91
7.6.6 Conclusion Business Rules 92

8 Conclusion 93

9 Discussion 96

References 99

10 Appendix 104
10.1 MILP model 104
10.2 Unsuccessful Models 111
10.2.1 Gradient-Based Optimization 111
10.2.2 Symbiotic Organisms Search 112
10.3 Flexible time window results SS+LS 114

List of Figures

1.1 Overview of manual C2Pa procedure 2
1.2 Overview of criteria and C2Pa stakeholders 4

4.1 Distribution of consultants over the job positions 22
4.2 Distribution of consultants over the service lines 22
4.3 Distribution of team members per project 24
4.4 Distribution of total skills per project 24
4.5 Relation between team members per project and skills per project 24
4.6 Distribution of team member role utilization 25
4.7 Distribution of project durations 26

6.1 Project priority decoding 35
6.2 Start time decoding 35
6.3 Consultant decoding 36
6.4 Skills decoding 36
6.5 Two-point crossover 50
6.6 Reproduction BRKGA 50
6.7 Schematic overview BRKGA 53
6.8 Boxplots of objective values of hyperparameter sets BRKGA 56
6.9 Illustrative overview Scatter Search 59
6.10 Boxplots of objective values of hyperparameter sets SS 61

7.1 Convergence curve BRKGA vs BRKGA+LS of instance size 3 73
7.2 Convergence curve SS vs SS+LS of instance size 3 73
7.3 Boxplots of objective values of the models of instance size 3 74
7.4 Convergence curve BRKGA vs BRKGA+LS of instance size 6 76
7.5 Convergence curve SS vs SS+LS of instance size 6 76

7.6 Boxplots of objective values of the models of instance size 6 77
7.7 Convergence curve BRKGA vs BRKGA+LS of instance size 12 79
7.8 Convergence curve SS vs SS+LS of instance size 12 79
7.9 Boxplots of objective values of the models of instance size 12 80
7.10 Convergence curve BRKGA vs BRKGA+LS of instance size 24 82
7.11 Convergence curve SS vs SS+LS of instance size 24 82
7.12 Boxplots of objective values of the models of instance size 24 83
7.13 Polynomial Regression Fit on Convergence Time SS 85
7.14 Convergence curves business rules experiment 92

10.1 Fitness curve GBO Best vs Average 112

List of Tables

3.1 Comparison of RCPSP approaches based on features 15
3.2 Comparison of RCMPSP approaches based on features 16
3.3 Comparison of MSRCPSP approaches based on features 17
3.4 Comparison of MSRCMPSP approaches based on features 18

4.1 Cost per hour per job position 22

6.1 BRKGA parameter grid 54
6.2 BRKGA tuning top 5 results 55
6.3 Scatter Search parameter grid 60
6.4 Scatter Search tuning top 5 results 61

7.1 Skill match verification results 64
7.2 Skill match C66 and C69 64
7.3 Required skill level and consultant skill level 64
7.4 Satisfaction verification results 65
7.5 Satisfaction scores of consultants 66 and 69 65
7.6 Hourly cost verification results 65
7.7 Verification client projects over internal projects 66
7.8 Verification full-time projects over part-time projects 67
7.9 Verification long-term projects over short-term projects 67
7.10 Results of MILP 70
7.11 Results of BRKGA 70
7.12 Results of Scatter Search 70
7.13 Results experiment instance size 3 73
7.14 Results experiment instance size 6 76
7.15 Results additional experiments instance size 6 78

7.16 Results experiment instance size 12 80
7.17 Results additional experiment instance size 12 81
7.18 Results experiment instance size 24 83
7.19 SS results experiment flexible time windows 87
7.20 Data description of business rules experiment 89
7.21 Results experiment business rules 89

10.1 SS+LS results experiment flexible time windows 114

1 Introduction

1.1 Problem Description

Bearingpoint is an independent management and technology consultancy company that offers services in three business units: Consultancy, Products and Capital. The company has a global consulting network and supports clients in over 70 countries. The Amsterdam office is organized into four departments: Customer & Growth, People & Strategy, Data & Analytics, and Technology. Each of these departments represents a crucial facet of BearingPoint's service offerings, allowing the company to address a wide array of client needs with specialized expertise. This research, conducted within the Data & Analytics department but applied to all departments of BearingPoint Netherlands (BE NL), sheds light on a challenge that affects not just this department, but the entire consultancy industry: the Consultant-to-Project assignment (C2Pa) problem.

The C2Pa problem originates from the fundamental nature of consultancy work and the interaction between client demands and the availability of skilled consultants. It entails assigning the appropriate personnel to specific projects. Currently, at BearingPoint, a dedicated team known as the OTL (Operational Team Lead) Team manages these staff allocations. This team relies heavily on accumulated experience and intuition to make informed decisions about consultant assignments on a recurring basis. Their work is crucial, as it directly impacts both client satisfaction and the company's operational efficiency. However, while functional, this approach has significant limitations that become increasingly apparent as the company grows and takes on more projects. The manual labor involved in this process is its primary drawback, as it is excessively time-consuming and lacks the capacity for reasonable long-term planning. The manual assignment process is visualized in Figure 1.1 (created by (1)). The procedure can be summarized as follows:

The procedure is initiated with a valid incoming resource request that is connected to one or more service lines. Depending on whether the needed service lines are clear, the OTL team selects the appropriate consultants based on common sense, consultant availability, and business rules. This can be done by a service-line-specific OTL member or by a team during the weekly OTL meeting. If an OTL member fails to find a suitable consultant, the selection process moves to the weekly OTL meetings. There, potential matches are discussed. If a match is found, the consultant is contacted for confirmation. Should no appropriate match be identified, even after these discussions, the incoming project is ultimately rejected. This process ensures thorough consideration of all options before rejecting any project. Addressing this complicated problem, while juggling multiple projects and diverse interests, is a complex balancing act. It requires significant time and intuition but often results in sub-optimal decisions.

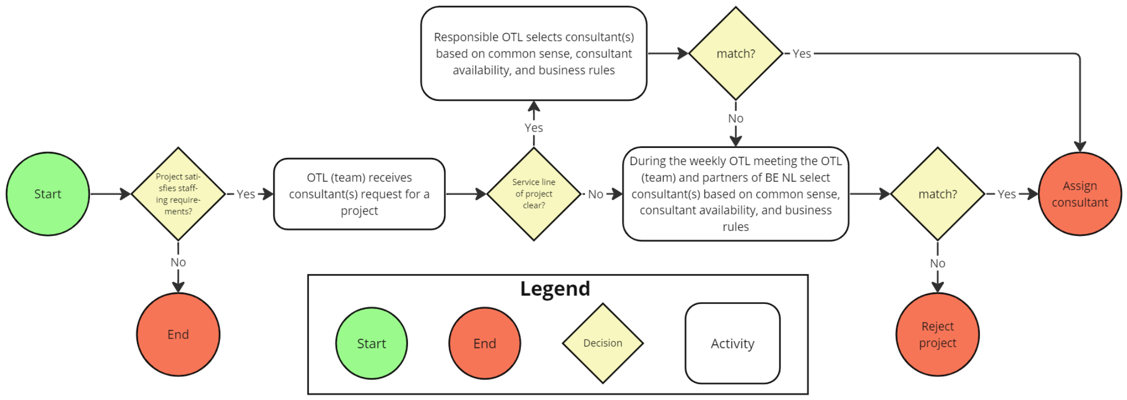
Start
Project satisfies staffing requirements?
Yes
No
End
OTL (team) receives consultant(s) request for a project
Service line of project clear?
Yes
Responsible OTL selects consultant(s) based on common sense, consultant availability, and business rules
match?
Yes
No
During the weekly OTL meeting the OTL (team) and partners of BE NL select consultant(s) based on common sense, consultant availability, and business rules
match?
Yes
Assign consultant
No
Reject project
Legend
Start
End
Decision
Activity

Figure 1.1: Overview of manual C2Pa procedure

The C2Pa process at BearingPoint can be triggered by various factors, with incoming project requests from clients being a primary driver. In addition, internal projects and initiatives also require careful allocation of resources. Regardless of the trigger, the objective remains consistent: to optimize resource allocation while balancing multiple, often competing, objectives and requirements.

The objective of this study, while seemingly straightforward, is more complex upon closer examination. It comprises five components, which we will refer to as Key Performance Indicators (KPIs). These KPIs help measure the success and efficiency of consultant allocation in project management. The first KPI, *Consultant Satisfaction KPI*, focuses on the satisfaction of the consultant, specifically the willingness to work on and enjoyment that is obtained by being assigned to the skills within projects. This KPI should measure the average satisfaction of all consultants assigned to projects based on the characteristics of the project and the wishes of the consultant. The second KPI in our framework is the *Skill Match KPI*. This indicator measures the alignment between consultants' proficiencies and the skill requirements of their assigned projects. The goal is to achieve a value as close to zero as possible for this KPI. This approach serves two main purposes: it ensures consultants have sufficient expertise for their projects while preventing the assignment of overqualified consultants. By targeting a near-zero Skill Match KPI, the allocation process aims to create an optimal balance where consultants are neither under-challenged nor overqualified. This balance is expected to improve project results, resource efficiency, but also in some sense consultant satisfaction.

The third KPI, *Hourly Cost KPI*, tracks the weighted average hourly cost of consultants assigned to accepted projects. This is related to the fact that more expensive consultants are more experienced and have more skills. This should balance the assignment of experienced and less-experienced consultants. The fourth KPI, *Utilization Rate KPI*, is self-explanatory; however, it is focused on the utilization of consultants in client projects, as these drive revenue for BearingPoint. As a result, external projects will have priority over internal projects. Lastly, the number of declined projects acts as a penalty mechanism. This penalty is designed in such way that the priority is to accept as many projects as possible. Figure 1.2 (created by (1)) gives a clear overview of the C2Pa stakeholders and their needs, and presents how different KPIs relate to the business setting.

Optimizing the aforementioned KPIs must be done within the constraints of the consultants' available net working hours. This optimization process is further complicated by the fact that accepting projects reduces the remaining availability of consultants, potentially leading to declining a portion of incoming project requests. Therefore, it requires careful consideration of resource allocation, project prioritization, and long-term consultant availability.

The Data & Analytics department of BearingPoint has done a considerable amount of research on the C2Pa problem already. The goal of previous research was to make the C2Pa problem structured and unambiguous. As part of the research, a tool was made that could solve the C2Pa process to optimality, to be more precise an exact algorithm in the form of a Mixed-Integer Linear Program. The research of Zentveld (1) concluded that it is possible to make the C2Pa procedure structured, ambiguous, objective, balanced for all

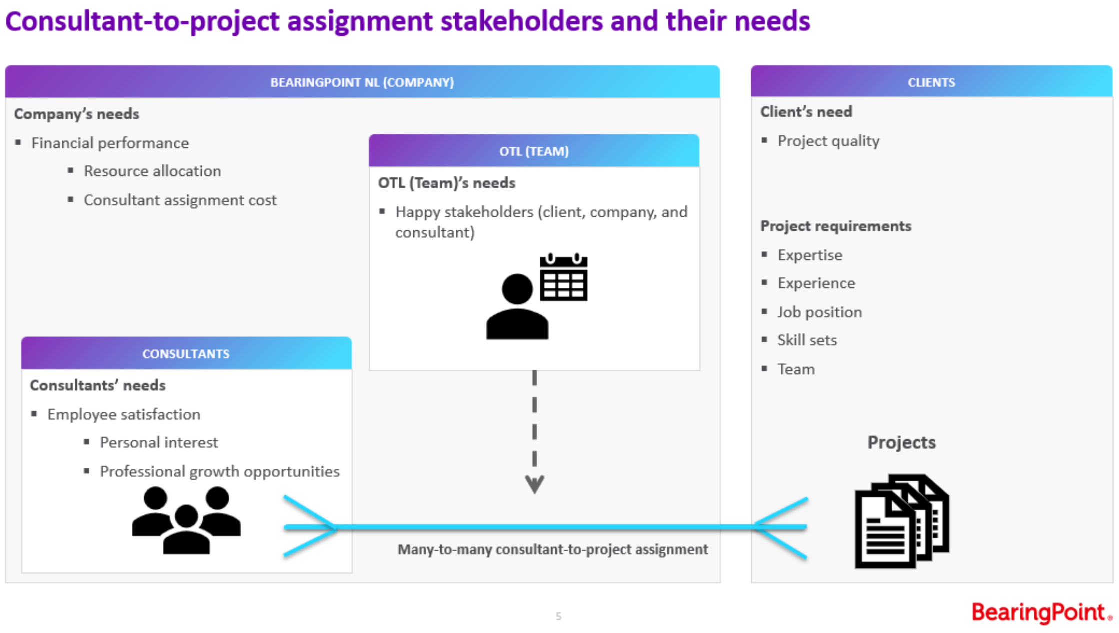

Figure 1.2: Overview of criteria and C2Pa stakeholders

involved stakeholders, transparent, and enables long-term planning. The performance of the model is quite impressive, as small instances, e.g. instances of 15 to 23 consultants and 9 to 14 projects, can be solved to optimality within reasonable time. However, Zentveld (1) also indicates that the relationship between instance size and computational effort has an exponential trend. This will cause serious feasibility problems when anticipating on expanding the team or in most ideal scenario running the model over the whole Dutch practice, consisting of approximately 100 consultants. This scalability issue is a critical concern for practical implementation. As the number of consultants, projects, and variables grow, the computational complexity of the problem increases. Consequently, finding optimal solutions becomes increasingly time consuming, potentially causing the algorithm to be impractical for real-time decision-making in larger instances, as the computational time increases exponentially with increasing problem size.

This problem opens up the opportunity to explore the field of non-exact optimization algorithms, called meta-heuristics. This field includes techniques that aim to find good, but not necessarily optimal, solutions to complex optimization problems where exact methods face significant challenges. Since these algorithms are very computationally efficient compared to exact methods, this research will be centered on the implementation of meta-heuristics combined with local search variants on the C2Pa problem. Meta-heuristics contain hyperparameters that significantly influence their performance and efficiency. The tuning of these hyperparameters is a critical aspect, as it directly impacts the algorithm's ability to find optimal or near-optimal solutions. A thorough tuning procedure will be conducted, following a systematic approach informed by existing research, to optimize parameter values. The meta-heuristics will be compared to the MILP model of Zentveld (1) where possible, and additionally, they will be exposed to multiple experiments. These experiments will be based on a data instance that contains the information of the entire BearingPoint NL practice. This implies that the base data instance is already significantly larger in terms of consultants and number of different skills compared to the research of Zentveld (1), but initially smaller in terms of projects.

1.2 Relevance of the problem

The consultant-to-project assignment (C2Pa) problem is a crucial business process for BearingPoint. As a consulting company, efficiently assigning consultants to client projects is essential for BearingPoint's operations and financial performance.

The C2Pa process involves strategically matching the needs of BearingPoint with the needs of its clients. From the company's perspective, efficient resource allocation and low consultant assignment costs are vital for profitability and cost-effectiveness. In addition, ensuring consultant satisfaction and providing opportunities for employees are crucial to maintaining a motivated and skilled workforce.

On the other hand, clients expect high-quality project delivery, which requires assigning consultants with the appropriate skills and expertise to their projects. Failure to do so can result in dissatisfied clients, damaged reputation, and the possible loss of future business opportunities.

Currently, BearingPoint relies on a dedicated team to manage project staff assignments manually, using accumulated experience and intuition. However, this manual process is time-consuming, lacks the ability to plan reasonably far ahead, and becomes increasingly challenging as the company expands and takes on more projects. By developing an efficient C2Pa model, BearingPoint aims to streamline the consultant-to-project assignment process, reduce the manual effort involved, and enable more proactive planning. The primary objective of this research thesis is to investigate the feasibility of implementing efficient meta-heuristic techniques for the practice-wide C2Pa problem that could be utilized as a decision support tool by the OTL team.

The proposed meta-heuristic models of this research could help BearingPoint in multiple ways. First, the models should be adaptive to the needs of the company, that is, changing preference in strategic objectives should be reflected in the resulting solutions. Second, the models could give strategic insights to the C2Pa procedure by, for example, demonstrating what the effect is of having a more flexible starting date are in terms of strategic objectives. Third, it should give the OTL team the opportunity to adress the assignment of consultants on multi-service-line projects, as cooperation between different service-lines is common within the consultancy industry.

1.3 Research Questions

Based on the description of the problem and the relevance of the problem, the following main research question can be stated:

Does the implementation of local search boost the performance of meta-heuristics in solving the Consultant-to-Project Assignment (C2Pa) problem at BearingPoint NL, specifically focusing on efficiency and efficacy?

Other questions that will be investigated in this research are:

- *What are the effects on the generated solutions when increasing the instance size of the problem and what are the limits of the models?*
- *What are the effects on the generated solutions when varying the starting time windows of incoming projects?*
- *What are the effects on the generated solutions in terms of convergence speed and solution quality when implementing business rules that restrict the availability of consultants?*

1.4 Research Outline

The remainder of the research is outlined as follows: in Chapter 2, the limitations and requirements related to the C2Pa procedure will be explained. In the next chapter, Chapter 3, the C2Pa problem is compared to existing research and its place within the current literature is established. In addition, the foundational problem base will be established, succeeded by a comprehensive review of existing meta-heuristic models. The insights derived from this analysis will inform the selection of models to be employed in the subsequent implementation phase. Chapter 4 presents an explanation and an analysis of the data. This will be followed by Chapter 5, which will describe the existing Mixed-Integer Linear Program of Zentveld (1), covering its variables, objective function, and constraints.

Chapter 6 presents the methodology, including a comparison of different algorithmic approaches. This chapter will detail the solution representation, the decoding procedure, the objective function, the local search approaches, and two meta-heuristic methods: BRKGA with Shaking (and Local Search), and Scatter Search (with Local Search). The hyperparameter tuning process for each method will also be discussed. Chapter 7 will focus on the evaluation of the results. This chapter will cover the verification and validation of models, a comparison to OTL and MILP, experiments on instance size, flexible time windows, and business rules. The evaluation will provide a comprehensive analysis of the methodologies presented in Chapter 6. In these experiments, the impact of local search will be closely examined by testing two versions of the models: one with local search implemented and one without local search.

Chapters 8 and 9 present the conclusion and discussion of the research. These chapters will critically analyze the results in relation to the research questions, discuss the study's limitations, and explore its implications. The conclusion will summarize key findings, assess the achievement of research objectives, highlight contributions to the field, and suggest directions for future research. Lastly, the research is finalized with an Appendix, which can be found in Chapter 10. Here, additional results and two unsuccessful models along with their corresponding problems are discussed.

2 Problem Context

The C2Pa procedure has to be executed while respecting some limitations. The limitations or constraints listed below must be respected in the models to match reality as much as possible.

- Non-preemptiveness: BE NL enforces a business rule that a project has to be executed without any interruptions. A consultant should therefore be assigned to a project from start to finish. This rule enables continuity for consultants and clients and reduces the scheduling complexity.
- Team member constraints: The assignments are restricted by multiple rules: It is not allowed to assign more than one consultant to a single team member role. Similarly, all team member slots of an accepted project have to be filled. It is not allowed to assign more than one consultant to a single skill, every skill within a project can only be performed by one consultant. Lastly, it is not allowed to switch skills within a project, a skill has to be executed by the same consultant from start to end.
- Time windows: Every request is obliged to submit a earliest and latest project starting time, which can be identical. The starting date of a project has to be planned within the interval. The strictness of this interval determines the availability of the consultants and by that influences the performance of the C2Pa procedure.
- Consultants availability: To preserve realism and respect the working conditions of the consultants, the OTL is only allowed to assign a consultant to a project if and only if the whole project fits within the working hours of a consultant.

Besides constraints, the characteristics of the C2Pa problem connected to the complexity also require an explanation. This will be referred to as the requirements, as the proposed models should incorporate these items in order to meet the objectives of the C2Pa problem.

- Declining a project: According to BE NL, a project can only be declined if assigning consultants would lead to violations of any kind.
- Scheduling: In the context of the C2Pa problem, scheduling is defined as determining the starting time of an accepted project.
- Assignment: In the context of the C2Pa problem, assignment is referred to as matching consultants to projects. The assignments are many-to-many as multiple projects have to be filled with consultants, where consultants can be assigned to multiple projects in case of part-time projects.
- Simultaneous scheduling and assignment: Combining scheduling and assignment makes it so that C2Pa is a simultaneous scheduling and assignment problem. This is a natural conclusion as the scheduling of projects directly influences the availability of the consultants and with that the possible assignments that can be made. Consequently, scheduling and assigning are dependent on each other.
- Utilization: BE NL prioritizes project-based learning for management analysts and consultants, aiming to maximize their project involvement. To meet annual utilization targets, the OTL team strives to allocate 100% of these employees' time to projects. This approach helps mitigate potential utilization losses that could occur when there is a gap between project assignments. By consistently aiming for full project allocation, the OTL team compensates for factors outside their control, ensuring optimal utilization rates.
- Project Priority: In the C2Pa process, BE NL prioritizes projects based on several key factors to optimize consultant utilization and maintain financial stability. The priority order is as follows:
  - Signed projects take priority over unsigned, non-chargeable work, or future reservations. This ensures commitment to existing contracts.
  - Existing client engagements and extensions are prioritized over new client projects, fostering ongoing client relationships.

- Full-time assignments are favored over part-time projects. This simplifies the C2Pa procedure and makes it easier to meet utilization targets.
- Long-term projects are preferred over short-term ones, as they provide more financial stability for BE NL.
- Client projects are prioritized over internal projects to generate revenue, which is crucial for the company's financial health.

By incorporating these constraints and priorities, the C2Pa problem becomes more reflective of real-world scenarios and aligns closely with BE NL's objectives. In addition to this, the literature search becomes more targeted.

3

Literature Review

In this chapter, the existing literature will be consulted to discuss the problem base of the consultant-to-project problem (C2Pa). The discussion conducted in Section 3.1 will cover the considerations that led to the existing MILP model (1). In that section, different variants of the main problem will be discussed, and the gaps between C2Pa and these variants will be highlighted. The limitations and requirements mentioned in Chapter 2 will be used to make judgements about each extension. Once the problem base has been set, the emphasis will shift to the research field of meta-heuristics, which will start in Section 3.2. There, the application of meta-heuristics on the different variants of the problem base will be discussed, the differences between the C2Pa problem base and the problem base of the proposed papers, the applicability of the proposed models, the performance of the proposed models, and relevant and/or unique elements of the proposed models will be discussed.

3.1 Problem base

The constraints and requirements described in Chapter 2 will be used as the basis of the literature search. As described in Chapter 2, the problem is two-fold, consisting of a scheduling and a staffing phase. For C2Pa, the scheduling phase corresponds to determining the starting time of the projects. Each project has its own requirements, while competing with each other for the availability of the consultants, which can be seen as limited resources. In 1964, Dike (2) standardized this problem as the Resource-Constrained Project Scheduling Problem, abbreviated RCPSP. Rinnooy Kan et al. (3) have shown that RCPSP belongs to the class of NP-hard problems. Briskorn and Hartmann (4) summarize RCPSP as follows: A project consists of J activities, labeled from 1 to J. Each activity j has a duration d_j and, once started, cannot be interrupted. Activities have precedence relationships, defined by sets of immediate predecessors P_j. An activity j can only begin after all its predecessors $i \in P_j$ are completed. Resources, labeled k = 1 to K, are required for activities. Each resource k has a per-period availability R_k. Activity j requires r_{jk} units of resource k during its execution. Two additional dummy activities, 0 and J+1, represent the project's start and completion respectively. These have zero duration and no resource requirements. The model assumes all information is deterministic and known in advance, allowing for precise planning and scheduling of activities within resource constraints and precedence relationships.

The RCPSP model is not enough to fully comply with all the specifications of C2Pa. Firstly, the RCPSP only considers a single project, whereas C2Pa needs to schedule a project portfolio. An extension of RCPSP, the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP)(5), considers this scenario. Secondly, in the C2Pa problem, there are multi-skilled consultants, which RCPSP does not have. The Multi-Skill Resource-Constrained Project Scheduling Problem (MSRCPSP)(6) is capable of representing the multi-skill workforce and at the same time scheduling the corresponding activities. An extensive review of the literature of this extension can be found in the paper of Afshar-Nadjafi (7). Combining these two extensions gives the final extension: the Multi-Skill Resource-Constrained Multi-Project Scheduling Problem (MSRCMPSP). The purpose of MSRCMPSP models comply with the goal of C2Pa, as a model is needed that is able to simultaneously schedule the projects and assign the multi-skilled workforce to the project activities.

However, some fundamental aspects inherited in MSRCMPSP should not be present in C2Pa, and some fundamental aspects of C2Pa are not present in MSRCMPSP models. An MSRCMPSP model considers the concept of tasks, the scheduling of tasks, and the precedence relation between tasks within a project. This is different compared to C2Pa, where the projects need to be scheduled and the consultants need to be assigned to skills based on their availability, skill level and satisfaction score. The proposed model of Heimerl and Kolisch (8) considers a simultaneous multi-project scheduling and multi-skilled staff assignment model which solves the following problems and introduces the following concepts: removes the concept of tasks, introduces internal and external projects, and is built on the assumption that a consultant is assignable to multiple projects at the same period. Although valuable, the model proposed by Heimerl and Kolisch (8) required further modifications to more accurately reflect the C2Pa process. Drawing inspiration from Zhu et al. (9), an adaptation that imposes limits on project team sizes was implemented. This enhancement was achieved by introducing a new parameter, N_{jk}, which specifies the prescribed number of consultants to be assigned to task j of project k. Other extensions are connected to the KPIs of the problem, which will be discussed in more detail in Section 5 and 6.

3.2 Application of Meta-heuristics

The exploration for effective models starts from insights gained in the RCPSP research area. By building upon the findings regarding RCPSP and extending the search to the other variants, we aim to identify models that can be adapted or enhanced to better suit the unique requirements of the C2Pa process. After each subsection, a table is presented that compares the features of the approaches. In this table, the abbrevations MO, MS WF and MP represent multi-objective, multi-skill workforce, and multi-project. Note that the emphasis will primarily be on the algorithmic features of the models and their performance compared to competitors, rather than the strength of resemblance to C2Pa. This decision has been made since the representation of the problems are quite similar and the emphasis is more on the capability of finding (close to) optimal solutions in a complex search space.

3.2.1 RCPSP

The Resource-Constrained Project Scheduling Problem is the standard problem and can be extended to the other variants. Consequently, a lot of research has been done on this problem. To give an overview of the latest techniques, Pellerin et al. (10) conducted a survey of hybrid meta-heuristics applied to RCPSP. These algorithms combine meta-heuristics with a type of schedule improvement procedure. In total, 36 methods are compared on the standard test sets of the library PSPLIB (11). Debels and Vanhoucke (12) and Paraskevopoulos et al. (13) propose two promising models.

Debels and Vanhoucke (12) deployed a *Bi-population Genetic Algorithm* (BPGA) to minimize the makespan of a project. With this approach, they managed to achieve the highest performance in multiple instances. The problem has been translated into the Activity List representation. They used 2-point crossover based on a 2-tournament parent selection procedure on parents of one population, and fed the children to the other population. In the context of genetic algorithms, parents are referred to as individuals whose genetic material is combined in the recombination step, while children are the offspring that follow from the combination of parents. The characteristics of the populations are distinct, since one is left-justified and the other right-justified. A conditional diversification method was used to avoid a homogeneous population. Additionally, forward/backward scheduling was used as a local search procedure. The survey suggests that BPGA has superior performance combined with excellent schedule computation speed.

Paraskevopoulos et al. (13) deployed a *Scatter Search* algorithm with *Adaptive Iterated Local Search*, shortened to *SS-AILS*. In the same survey, this model was ranked among the top performers. The Event-List representation has been used in the evolution process. The algorithm uses reference sets that operate on the event-list representation, and consists of an initial phase, in which diverse solutions are built, and a scatter search phase. The fact that it uses reference sets indicates that the algorithm is population-based. SS-AILS has two components, an adaptive perturbation strategy that modifies the current solution, and a local search procedure that consists of compound moves and neighborhood structures which are based on memory. The performance of this algorithm is excellent, however, it struggles speed-wise when the instances become significantly larger.

The algorithm proposed by Almeida et al. (14), while not covered in Pellerin's review, offers a flexible implementation that is advantageous for addressing the potentially complex extensions of C2Pa. This approach employs a *Biased Random Key Genetic Algorithm* with a Random-Key representation, encoding the chromosome as a vector of uniform random numbers between 0 and 1. A chromosome is a representation vector that's specifically tailored to the genetic algorithm's problem domain. This representation translates the problem into a priority-based permutation, enabling manipulation by evolutionary operators. The algorithm incorporates an elitist selection procedure and a uniform crossover strategy that always pairs an elite parent with a non-elite one. Crossover is simply the process of exchanging genetic material between two parent chromosomes to create offspring. Additionally, it replaces a percentage of the worst-performing individuals with randomly generated mutant vectors. Despite the challenges of direct performance comparisons due to differing test instances, the algorithm demonstrates notable ability to find near-optimal solutions, suggesting its potential effectiveness for C2Pa applications.

Table 3.1 provides a comprehensive overview of the approaches discussed above. It should be noted that RCPSP models do not align with the specifications of C2Pa in any of the categories. However, the model specifications reveal that implementation allows for various problem-solving strategies. Some representations are more tailored to specific problems,

	Algorithm	Representation	MO	MS WF	MP	Precedence	Project selection	Skill division	Population
Debels & Vanhoucke	BPGA	Activity-List	×	×	×	✓	×	×	✓
Paraskevopoulos et al.	SS-AILS	Event-List	×	×	×	✓	×	×	✓
Almeida et al.	BRKGA	Random Key	×	×	×	✓	×	×	✓
C2Pa	-	-	✓	✓	✓	×	✓	✓	-

Table 3.1: Comparison of RCPSP approaches based on features

while others, such as Random-Key, demonstrate versatility across a wide range of problems. The analysis suggests that a population-based algorithm, complemented by intensification and diversification procedures, appears to be the most promising approach to address the RCPSP. To bridge the gap between RCPSP models and C2Pa requirements, the subsequent sections will explore extensions that should offer additional specifications in their models. These extensions aim to address missing model specifications, potentially leading to a more suitable solution for the C2Pa problem.

3.2.2 RCMPSP

The first paper by Goncalves et al. (15) introduces a *Biased Random Key Genetic Algorithm* designed to address the RCMPSP. The problem is encoded using a random key-based chromosome representation that includes priorities, delay times, and release dates. The values in the random key vector correspond to the priority of the activities. A heuristic approach that uses the priority values of the chromosome generates the so-called active schedules. Identical to the approach of (14), a biased reproduction procedure is implemented combined with an elitist survival strategy and a mutant replacement strategy used as mutation. The parameterized uniform crossover was implemented opposed to the frequently used 1 or 2-point crossover.

Bredael and Vanhoucke (16) follow a similar algorithmic design by creating a *Genetic Algorithm.* The GA is driven by the random key chromosome representation. Several schedule generators are implemented that are based on either the random key vector, priority rules, measurements, or scheduling techniques. After extensive experiments, they concluded that the best configuration consists of 2-point crossover, a high mutation rate and a tournament-fitness parent selection operator. In their performance analysis, the proposed GA achieved superior performance over ten other meta-heuristics in a fifth of the instances.

Table 3.2 presents an overview of the comparison between C2Pa and the two discussed papers. Adding the multi-project aspect to the problem gives no indication on how project

	Algorithm	Representation	MO	MS WF	MP	Precedence	Project selection	Skill division	Population
Goncalves et al.	BRKGA	Random Key	×	×	✓	✓	×	×	✓
Bredael & Vanhoucke	GA	Random Key	×	×	✓	✓	×	×	✓
C2Pa	-	-	✓	✓	✓	×	✓	✓	-

Table 3.2: Comparison of RCMPSP approaches based on features

selection should be handled, since the papers only address the scheduling of the activities within the projects. Moreover, the presence of precedence conflicts with the requirements of C2Pa. The papers indicate that any kind of GA combined with the random key representation should be best suited for the problem. Goncalves et al. (15) and Bredael and Vanhoucke (16) explain that the choice for the RK-representation was made because it showed convergence capability, has an effective solution structure, it is adaptable to various problem constraints, it allows flexibility in schedule generation, and it suits the genetic algorithm framework perfectly. Despite the absence of most of the requirements of C2Pa, like multi-skilled workforce, skill division and project selection, this section has given great insights on a promising representation and evolutionary framework.

3.2.3 MSRCPSP

The Multi-skill/mode Resource-Constrained Project Scheduling Problem extends the base problem by introducing either a multi-skilled workforce or multiple execution modes. It has been decided to merge these two variants in one section as the notions share similarities. The multi-mode extension is an active research field. For example, the instance test dataset PSPLIB by Kolisch and Sprecher (11) is often utilized to benchmark promising algorithms. Another benchmark is the MMLIB instance set by Van Peteghem and Vanhoucke (17). In the same paper, an investigation is executed that compares meta-heuristics on these instances.

Maghsoudlou et al. (18) explored multiple algorithms to solve the multi-skilled RCPSP. A bi-objective problem was solved using multiple random key vectors where the solutions were ranked using the non-dominated ranking method. In their problem, each activity needed one worker and at least one skill, indicating that skill division is included. The proposed random key representation ensured feasibility. An analysis concluded that the *Fuzzy Sorting Cuckoo Search* algorithm, which employed an invasive weeds algorithm, was the best algorithm overall.

Machado et al. (19) proposes a *Genetic Algorithm + Variable Neighborhood Search* algorithm that solves the multi-mode RCPSP. The initial population is generated by randomly selecting priority rules. The variable neighborhood search consists of two neighborhood moves that act on the activity-list representation. Moreover, a uniform crossover and swap mutation operator were implemented. In terms of performance, this algorithm does not show significant statistical differences in percentage deviation compared to the top four meta-heuristics in the PSPLIB benchmark. Additionally, it outperforms state-of-the-art algorithms in small- and medium instances.

In the experimental investigation of Van Peteghem and Vanhoucke (17), the *Scatter Search* algorithm of Van Peteghem and Vanhoucke (20) noted the most impressive performance in large instances. The scatter search algorithm makes use of a random key solution vector, which determines the sequence of activities, and a mode list. After initialization, the population is split into two smaller reference sets, where one set contain good solutions and the other diverse solutions. These sets are submitted to a subset generation method in which pairs of solutions are made for the recombination procedure. The 2-point crossover procedure was found to be the most successful. Last, a cycle of improvement methods and two local search methods is applied to the solutions, after which the reference sets need to be updated. The algorithm not only exhibits outstanding performance, but also proves to be highly efficient.

	Algorithm	Representation	MO	MS WF	MP	Precedence	Project selection	Skill division	Population
Maghsoudlou et al.	FSCS	Random Key	✓	✓	×	✓	×	✓	✓
Machado et al.	GA-VNS	Activity-List	×	✓	×	✓	×	×	✓
Peteghem and Vanhoucke	SS	RK+ML	×	✓	×	✓	×	×	✓
C2Pa	-	-	✓	✓	✓	×	✓	✓	-

Table 3.3: Comparison of MSRCPSP approaches based on features

Table 3.3 presents an overview of the described algorithms of MSRCPSP. The algorithm proposed by Maghsoudlou et al. (18) showed the most similarities with C2Pa until now. The key finding of their algorithm is the usage of random key vectors. This representation allowed for the flexibility to encode the skill division aspect in combination with the multi-skilled workforce. In line with previous findings, the other models combine methods to include intensification and diversification. The scatter search algorithm has both aspects built into the algorithm, as the diversification is implied by reference set 2 and intensification by the combination of two solutions from reference set 1. This algorithm partly uses the random key representation, which makes it feasible to implement the missing C2Pa specifications. The algorithm of Machado reiterates the importance of having a local search procedure. Despite none of the algorithms covering the multi-project and project selection specifications of C2Pa, the findings in this section provide significant insights on how to solve these problems.

3.2.4 MSRCMPSP

The Multi-Skill Resource-Constrained Multi-Project Scheduling Problem is an extension that offers the addition of consecutively scheduling multiple projects while dealing with multi-skilled workforce. The research regarding this extension is very limited. Nevertheless, it resembles C2Pa the most.

Chen et al. (21) propose a *Non-dominated Sorting Genetic Algorithm* to solve a multi-objective model for the MSRCMPSP implemented for IT project development while considering evolution of competency of workforce. Their goal was to generate a weighted ideal point of the approximate optimal Pareto solution set. The activity-list representation of the chromosome was used and single-point crossover and single-point mutation operators were applied to the encoding. Individuals are ranked based on Pareto dominance, where solutions dominating others receive lower ranks, forming non-dominated fronts. In addition to the crossover and mutation operators, no other intensification or diversification methods were implemented.

Lastly, the algorithm proposed by Torba et al. (22) will be discussed. They proposed a memetic algorithm that combine a *hybrid Simulated Genetic Algorithm* with a conditional simulated annealing as a local search procedure. It is a single-objective population-based model that uses both crossover and mutation operators. The initial solutions are constructed by a randomized priority rule procedure that is in the form of an activity-list. In their experiments, excellent performance is achieved both in small and large instances. In addition to that, the additional value of the conditional simulated annealing approach has been demonstrated.

	Algorithm	Representation	MO	MS WF	MP	Precedence	Project selection	Skill division	Population
Chen et al.	NSGA-II	Activity-List	✓	✓	✓	✓	×	×	✓
Torba et al.	hSGA + SA	Activity-List	×	✓	✓	✓	×	×	✓
C2Pa	-	-	✓	✓	✓	×	✓	✓	-

Table 3.4: Comparison of MSRCMPSP approaches based on features

Table 3.4 provides a comprehensive overview of the approaches discussed above. Although MSRCMPSP is most consistent with C2Pa, all the models discussed still lack the components project selection and skill division among the workforce, which are prominent parts of the C2Pa procedure. As in the other sections, precedence is included in every model. Moreover, a multi-objective algorithm was proposed that made use of a non-dominated selection procedure. The model provided by Torba et al. (22) seems the most promising, as it includes intensification and diversification components that have been proven to add value to the main problem. Although covering many specifications of C2Pa, the activity list representation appears to be infeasible to encode the skill division aspect of the C2Pa problem.

3.3 Conclusion

The discussions in Sections 3.2.1-3.2.4 indicate that significant research has been done on RCPSP and the extensions. However, none of the discussed algorithms match the specifications of the C2Pa problem. The RCPSP models lack the presence of multi-objectiveness, multi-skilled workforce, multi-project, project selection, and skill division. The RCMPSP models lack the presence of multi-objectiveness, multi-skilled workforce, project selection, and skill division. For MSRCPSP, the models lack multi-project and project selection. Lastly, the MSRCMPSP models lack the presence of project selection and skill division.

In summary, the existing RCPSP extensions cannot match all the specifications of the C2Pa problem, which results in needed solutions for the following gaps.

- Propose a method that implements project selection
- Propose a method that is not built on precedence relations
- Propose a multi-objective method that is not related to the starting or finishing times of projects

Another problem is the fact that the solutions to the specifications do not all originate from the same algorithm. Consequently, a mismatch in the implementation strategy will occur when trying to fit all solutions into one algorithm. This requires choosing a representation that is flexible and capable of including all specifications, including the additional specifications that could be added in the experimental phase of the research.

From the research, it can be concluded that the random key vector representation is a promising candidate, as it is proposed in multiple high-performing models because of its flexibility and capabilities. Moreover, Maghsoudlou et al. (18) managed to incorporate skill division by using this representation. It is evident that population-based algorithms are the most suitable for addressing the C2Pa problem. These algorithms have shown promising results in the literature, they are robust, are capable of exploring different parts of the search space, and can be combined with other optimization techniques. Specifically, we will focus on two variants of genetic algorithms: the scatter search and the biased random key genetic algorithm (BRKGA). These algorithms incorporate both intensification and diversification procedures. Intensification is achieved through methods such as local search and crossover, while diversification is implemented through mutation or other disruptive techniques. Combining a flexible random key vector with effective algorithmic methods while taking the efficiency into account should result in a high performing meta-heuristic that solves the C2Pa problem.

Moving forward, the role of local search will be a central point of the research. While local search methods have demonstrated their effectiveness in improving solutions, their potential to negatively impact computational efficiency requires careful consideration. Given their ability to explore the solution space more thoroughly and potentially find higher-quality solutions, incorporating local search techniques could be beneficial. However, it is essential to balance these advantages with the potential computational overhead that are associated with it. The aim is to investigate the extent to which local search is crucial in solving the C2Pa problem, evaluating its efficiency, effectiveness, and potential limitations. This will be done by enabling and disabling the local search procedure within each algorithm for each experiment. The performance metrics resulting from the experiments will be analyzed in order to draw conclusions.

4

Data

The data needed as input for the algorithms is stored in three locations within the data storage of BE NL. The data can be categorized in three categories, Consultant Capability Data, Consultant Availability Data, and Project Data. In this chapter, the data that falls under each category is allocated a section in which the data will be explained and analyzed.

4.1 Consultant Capability Data

Under consultant capability data, all the attributes of the consultants regarding consultant skills are stored. This data has been internally collected by means of surveys. In the survey, a consultant is asked to provide a ranking of their satisfaction score and skill level on a specific skill. The list of skills is multi-department and contains skills about hard skills, soft skills, and general skills. Because consultant work can be multidisciplinary, it is allowed to rank skills that are not within the main department of a consultant. The satisfaction score, stored in the variable css, ranges from 1 to 10. A skill gets a satisfaction of 1 if the consultant does not rank a skill. The skill level, stored in the variable csl, ranges from 0 to 3, where 0 corresponds to not having knowledge of the skill. In total, 127 unique skills have been identified by BearingPoint. Analysis of the surveys indicates that each consultant possesses 21 skills on average. This number is heavily distorted by the difference in job level, as consultants with a senior job position usually have developed more skills than a newly acquired management analyst.

Other variables that are stored are the names of the consultants, the job position of consultants, and the service line of consultants. Figure 4.1 shows the distribution of the consultants over the job positions. This shows that within BE NL six different job positions are possible, ranging from management analyst to senior manager. It can be said that the higher job position, the more consultants, aside from the senior manager position. Figure 4.2 shows the distribution of consultants across the service lines. The service lines D&A and TECH are clearly underpresented compared to the larger service lines C&G and P&S.

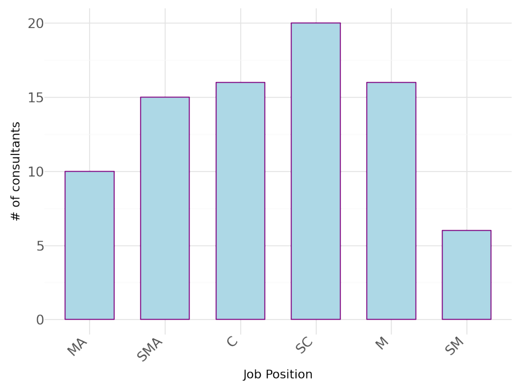

Figure 4.1: Distribution of consultants over the job positions

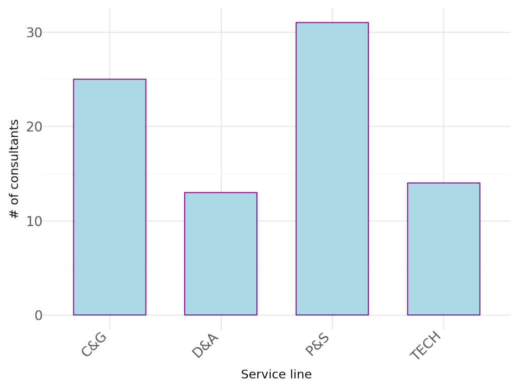

Figure 4.2: Distribution of consultants over the service lines

Table 4.1 presents the hourly cost, stored in the variable c, of each job position. The cost of a senior manager has been inflated, as this position is rarely assigned to a project as a team member. Because of that, the algorithms will possibly refrain from assigning this position to a project.

Job position	Cost per hour (€)
Management Analyst	115
Senior Management Analyst	120
Consultant	127
Senior consultant	140
Manager	165
Senior Manager	220

Table 4.1: Cost per hour per job position

4.2 Consultant Availability Data

Consultant availability is related to the weekly contractual working hours and the weekly net available working hours. Generally, the contractual weekly working hours (WH) are 40 hours. For simplicity, it is assumed that every consultant works 40 hours a week.

The weekly net available hours (NH) of consultants is calculated from a file provided by a BearingPoint planning tool. This tool stores the events that deduct utilization from working hours, such as project assignments or holidays. In such an event, the time interval and the utilization percentage are stored. The file contains the events from the start of January 2024 until the end of 2025. In this research, a subset of this dataset will be used that contains events from Oktober 2024 to Oktober 2025. The first week of Oktober will thus be used as t=0. This implies that the availability of the first two months is relatively low, since most consultants are still working on ongoing projects in this time-interval. When their current projects are finished, almost every consultant should have full availability. The following procedure is done to calculate the weekly net available hours. First, the daily net available hours are calculated for each consultant. This is done by grouping the events of a specific consultant and subtracting the utilization from the corresponding days given the utilization period. This daily data is then transformed into weekly data to obtain the weekly net available hours.

Despite the transformations of the data, the quality of the data is still lacking. The names of the consultants in the capabilities data and availability data have different formatting. Moreover, the sets of names are also not identical. This originates from the fact that not all consultants have completed the survey. Alternatively, the planning tool does not include information on newly hired BE NL consultants. To fix this, first the formatting of the names has been fixed, and second the intersection of both consultant name sets has been determined. The intersection is then used to create a subset of the data. This transformation resulted in having a total of 74 consultants. The processed consultant data is now suitable for input into the algorithms.

4.3 Project Data

Project data contains information about the duration, team members, start date interval, required hours, and included skills with their required level of skills of projects. OTL members from the four service lines D&A, TECH, C&G, and P&S were asked to submit information about recent or past projects in a tool. In total, 24 projects were submitted, where each OTL member submitted a similar number of projects to ensure a balanced portfolio. The characteristics of the projects are quite different. Figure 4.3 shows the distribution of team members per project. In this figure, it can be seen that 50% of the projects require only one consultant. Beyond this, there is a clear trend where the frequency of projects decreases by roughly half for each additional team member required.

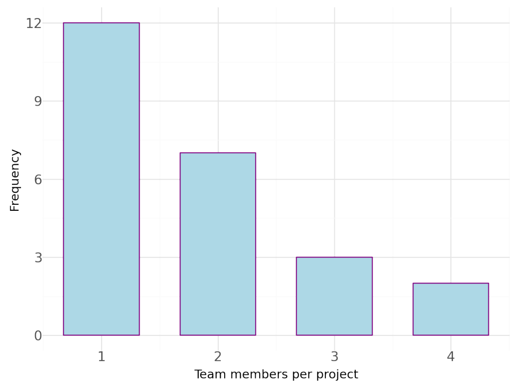

Figure 4.3: Distribution of team members per project

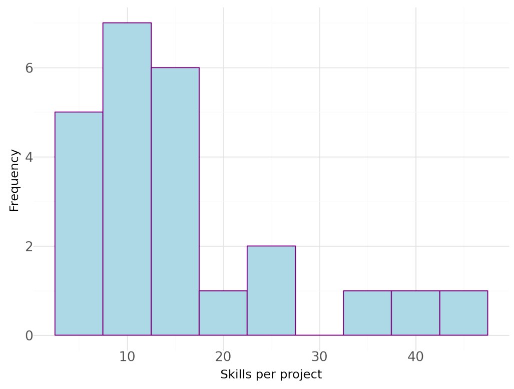

Figure 4.4: Distribution of total skills per project

Figure 4.4 shows the distribution of the number of skills involved in the projects. The majority of the projects require between 5 to 15 skills. However, there are projects that require between 35 and 50 skills. These are long-term projects that cover multiple service lines. Figure 4.5 shows the relation between the size of a project team and the number of involved skills per project. An overall increasing trend can be observed between team size and the number of skills required. However, this relationship is not consistent across all cases. Although larger teams often involve more skills, there are instances where increasing the number of team members does not necessarily lead to a corresponding increase in the skill set needed for the project.

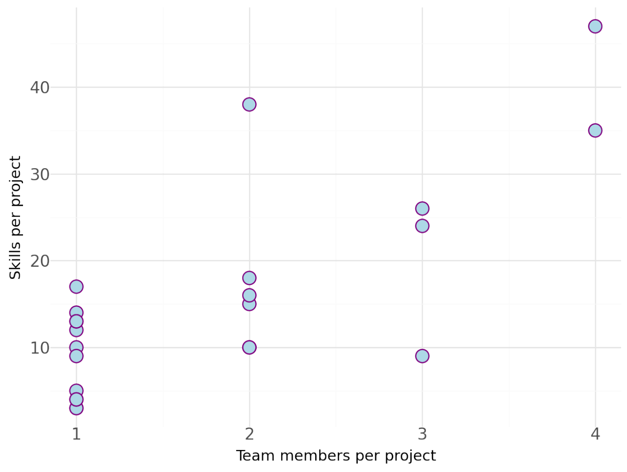

Figure 4.5: Relation between team members per project and skills per project

The skill level of the skills required for the projects have two levels, either basic knowledge is required, which requires a skill level of 1, or expert level knowledge is required, which requires a skill level of 3. This is done to make the process less complex for the OTL members. In practice, the possible skill levels can be extended to also incorporate skill level 2. The aspect that complicates the skill division is that the skills are not assigned to a specific team member of a project. This implies that the skills must be divided among the assigned consultants. The number of skills that a team member gets assigned is proportional to the scheduled utilization of a role. A consultant can only be assigned complete skills, fractional skills are not allowed to be assigned. To prevent fractional skills, a method has been implemented that adds a skill to the team member with the highest fraction. This will be done until the required number of skills has been divided. In addition to this, a minimal number of skills of 1 has been implemented.

Figure 4.6 shows the distribution of the scheduled utilization of the team member roles. It can be seen that utilization consists primarily of 100% and 80%, which means that consultants are invested primarily in only one project. The remaining project have varying utilization with small peaks at 10% and 40-50%, which enables consultants to work on multiple projects at the same time.

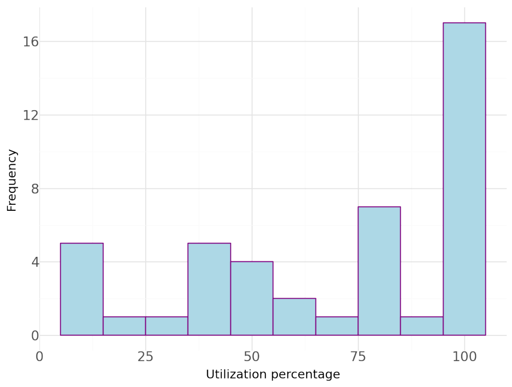

Figure 4.6: Distribution of team member role utilization

Figure 4.7 presents the distribution of the duration of the projects. The distribution is balanced, as all projects but one are evenly divided between 1 and 24 weeks. One project has a significantly longer duration of 36 weeks. The average project duration is 11.7 weeks, where the shortest project has a duration of 1 week. Lastly, the project portfolio is dominated by client projects. Of the 24 projects, only 4 projects are internal beach projects compared to the 20 external client projects. The average duration of the internal projects is 2 weeks, this indicates that internal projects are generally small assignments.

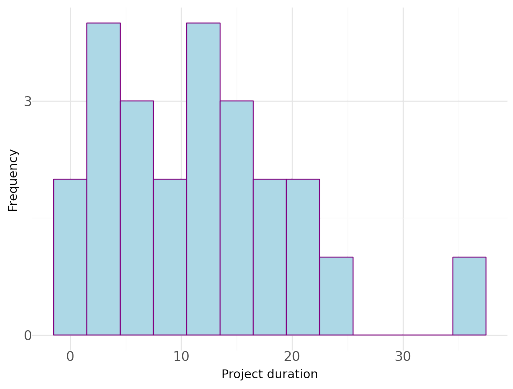

Figure 4.7: Distribution of project durations

The project data is derived from historical projects, with some projects dating from 2024 and others from 2023. In a realistic OTL scheduling setting, projects typically need to be scheduled in the near future following the project request. To simulate this scenario, the original starting times of the projects have been disregarded and replaced with randomly sampled starting dates ranging from 1 week to 12 weeks from October 1, 2024. This adjustment aligns project timelines more closely with a hypothetical current scheduling date. Furthermore, projects that originally had a flexible time window have been assigned a randomly sampled latest starting date, set within 1 to 2 weeks from their new earliest starting date. This modification preserves the concept of flexibility while constraining it to a more realistic short-term scheduling horizon.

5

Existing MILP model

In this chapter, the mathematical model of Zentveld (1) will be linked to the problem context described in Chapter 2. The complete mathematical model and the variable list can be found in Appendix 10.1. For each element present in the problem context, the corresponding part of the model will be referenced and explained. In addition, the most relevant aspects of this model will be covered. See (1) for a detailed explanation of each constraint of the model, the corresponding constraints will be referred to using square brackets.

Mixed Integer Linear Programming (MILP) models are optimization tools that combine continuous and integer variables to find the best solution within a set of constraints. By integrating linear programming with integer constraints, MILP can represent complex real-world problems involving discrete decisions. These models are widely used in fields such as supply chain management, production planning, resource allocation, and scheduling. Key components include an objective function, linear constraints, and decision variables, some of which must be integers.

Non-preemptiveness and Time windows Constraints [4.41], [4.46], [4.47], [4.49], [4.50], and [4.51] are implemented to enforce the critical business rule that a project must be executed without interruptions. These constraints can be categorized based on their functions in handling declined and accepted projects, as well as enforcing time windows.

For declined projects, three constraints play crucial roles. Constraint [4.41]restricts assignments when a project p is declined ($ap_p = 0$). Furthermore, constraint [4.46] uses the variable $u_{p,t}$ to ensure that a declined project p has not started at time t. The variable $pd_{p,t}$, which indicates whether a project p is executed at time t, is employed in constraint

[4.51] to confirm that there is no execution time allocated for a declined project.

For accepted projects, two constraints are essential. Constraint [4.49] ensures that an assigned consultant possesses precisely the number of skills specified by the variable $st_{p,m}$ when project p is executed at time t. Constraint [4.50] is implemented to guarantee uninterrupted project execution. It achieves this by equating the decision variable $pd_{p,t}$ to $u_{p,t}$ when the project starts ($u_{p,t} = 1$) and maintains $pd_{p,t} = 1$ until the project is completed ($t = t + pt_p$).

To ensure that projects are executed within the correct time interval, constraint [4.47] is incorporated. This constraint restricts the variable $x_{i,p,t,m,s}$ to only hold a value when time t falls within the range defined by the earliest starting date and the latest finishing date of the project.

Team member and skills constraints Constraints [4.42], [4.43], [4.44], [4.45], [4.48], [4.52], and [4.53] are responsible for enforcing the rules regarding team members and skills. These constraints can be categorized into two main groups: those managing team member roles and those managing skill assignments. Constraints [4.42], [4.43], and [4.44] enforce the rule that only one consultant is allowed per team member role. An auxiliary variable $z_{i,p,m}$ is introduced to indicate when consultant i is assigned to team member role m of project p. Constraint [4.42] ensures that only one consultant can be assigned per project and team member role. Constraint [4.43] guarantees that a consultant is assigned to only one team member role within a project. Constraint [4.44] ensures that assignments in $x_{i,p,t,m,s}$ are made only when $z_{i,p,m} = 1$, linking the assignment variable to the role allocation.

The remaining constraints manage skill assignments. Constraint [4.45] implements the rule that a specific skill of a project can only be assigned to one consultant. Constraint [4.48] enables the model to recognize the required skills ($rs_{s,p} = 1$), ensuring that only the skills required in project p can be assigned to a consultant. Constraints [4.52] and [4.53] implement the rule that skills must be assigned to the same consultant for the duration of the project. This is achieved by introducing the variable $v_{i,p,m,s}$, which equals 1 if consultant i is assigned to team member role m and skill s of project p. Constraint [4.53] forces the same behavior on variable x, thereby ensuring that the model assigns a skill of a project to a single consultant throughout the project's duration.

Consultants availability The availability of consultants is managed by a single, crucial constraint. Constraint [4.40] ensures that a consultant i does not exceed their available time in any given time period t (represented by $NH_{i,t}$). It is important to note that this constraint allows for flexible allocation of the remaining consultant time. Specifically, it enables consultants to work simultaneously on multiple projects, provided their total time commitment does not exceed their availability. This flexibility is achieved by summing all chargeable ($CH_{t,p,s}$) and non-chargeable hours ($NCH_{t,p,s}$) across all projects, roles, and skills to which the consultant is assigned ($x_{i,p,t,m,s} = 1$) for each time period t. In practice, the hours concerning engagement manager are included in the constraint, however, these are not implemented in the model.

KPIs and Deviations An extensive explanation of the calculations can be found in Chapter 6. In here, the differences in implementation between MILP and meta-heuristics is shortly addressed. The formulation of the cost KPI, the utilization KPI, the satisfaction KPI, and the skill match KPI can be found in constraints [4.36], [4.37], [4.38], and [4.39], respectively. The formulations contain two kinds of added variables. First of all, the constraints contain aspiration level variables in the form of a variable y. Constraints [4.66], [4.67], and [4.68] indicate what values the variables y can take. In addition to this, two kinds of deviation variables are introduced, d^+, d^- and e^+, e^-. These variables capture the positive and negative deviations between KPIs and continuous variables, and the positive and negative deviations between continuous variables and desired minimum or maximum aspiration levels of KPIs. For more information, see Appendix 10.1 and (1).

Objective function The objective function is stated in [4.35]. The objective function consists of the calculations of the KPIs, the deviations from the aspiration levels of the KPIs, and the total penalty for the declined projects. The contributions of the KPIs are stored in the parts that combine the KPI weights w^+ and w^- and the deviation variables d^+ and d^-. The penalties for deviations from the aspiration levels are stored in the variables e^+ and e^- with the weights α^+ and α^-. Lastly, the average project decline penalties is added. This part uses the penalty decline weight $w_{decline}$ and project priority value pv_p to determine the penalty of a single project. In practice, every weight and priority value is set to 1 so no KPI was given priority. Lastly, each variable connected to the KPIs is normalized by its own constant. The variable list in Appendix 10.1 elaborates on how these are calculated.

6

Methodology

In this chapter, the methodology of the proposed models will be explained. Initially, the modelling design will be covered, which includes the encoding and decoding procedure of the C2Pa problem. Afterwards, the definition and calculation of the KPIs of the objective function will be explained. Next, the procedure of two local search approaches is explained. This chapter will close with an in-depth outline of both the Biased Random Key Genetic Algorithm (BRKGA) and the Scatter Search (SS) models, in which the methodology and the hyperparameter tuning procedure will be explained.

6.1 Encoding

In this section, the C2Pa problem is formulated into a solution representation vector that serves as a framework for both models. To enhance efficiency, a method for reducing the solution search space has been developed. Translating the problem into a solution vector requires addressing the research gaps identified in the literature review. Section 6.1.1 will detail the search space reduction method. Section 6.1.2 will focus on addressing the research gap related to project selection and its corresponding encoding. The encoding process for consultant selection will be covered in Section 6.1.3. Section 6.1.4 will explain the encoding related to skill division. Lastly, Section 6.1.5 elaborates how the time windows are encoded into the chromosome. Collectively, this section will demonstrate how these elements are integrated into a cohesive solution representation vector.

Due to the requirements of the problem and the flexibility of the solution representation, it has been decided to use the random-key vector (RK-vector) as the solution representation. In the random-key representation, proposed by Bean (23), a vector consists of randomly generated real numbers in the interval [0,1), also called a chromosome, where each element represents the priority or rank of a decision variable. The order of the elements in the vector determines the relative importance or priority of the corresponding variables. This representation makes the method independent of the problem it tries to solve. Therefore, the random key presentation has been used in a range of optimization problems. In addition, this representation is suitable for evolutionary algorithms. The goal of the encoder is to translate the input data to a vector of real numbers, i.e., *input data* $\rightarrow [0,1)^Z$, where Z is the length of the random key vector.

6.1.1 Search Space Reduction

The search space for an algorithm that implements the random key method needs to be as small as possible to ensure efficiency and feasibility. A large search space exponentially increases the number of possible solutions, making it computationally expensive and time-consuming to find (close to) optimal solutions.

In order to reduce the number of possible solutions, an initial feasibility filter is applied to the input data that eliminates the projects that do not have sufficient available consultants to fill the activities of the projects. The availability of consultants is stored in the NH variable. Combining that with number of skills of an activity and the number of hours per skill of the project enables to check whether the cumulative time of the activity can fit in the available net hours of the consultant. This will be checked for every possible starting time in case of an interval of starting times. A consultant will be added to the availability list if it is available in any of the intervals from the start time to the finish time.

This procedure excludes projects from the project list when they are determined to be infeasible. Consequently, the information from this project regarding consultants and skills is also not included in the chromosome, therefore, shortening the size of the chromosome.

6.1.2 Project Selection

The project selection procedure is addressed through a collaborative effort between the encoder and the decoder. The encoding method for project selection is adapted from the approach used to translate activity priorities into the chromosome, as described by Goncalves and Resende (24). This concept of priority can also be effectively applied to projects. In the context of project selection, a higher priority indicates greater project importance. For a set of P projects, the encoding of the project priorities is represented by a vector of p random keys. In this vector, a lower value corresponds to a higher priority. Consequently, the project with the lowest priority value is considered first in the decoder.

This encoding strategy allows for a flexible and efficient representation of project priorities. Other advantages are: It inherently promotes diversity in the population, as chromosomes with identical random keys but different project priority orders can potentially generate distinct solutions. This diversity arises because the availability of consultants depends on the sequence in which projects are considered. The priority method effectively resolves conflicts between internal and external projects. In cases where a clash occurs, external projects should have a lower random value, ensuring they receive earlier consideration in the decoding process.

6.1.3 Consultant Selection

The procedure for selecting consultants follows a priority system, similar to the one previously employed for project selection. The feasibility filter is used to retrieve a list of available consultants of each project. The next step involves identifying and counting the available consultants from the projects that qualify for assignment. This count then determines the length of the consultant selection random-key vector. To be more specific, for a set of n eligible projects, where the consultants are gathered into a list denoted as N, the consultant selection process is encoded through a vector composed of $|N|$ random keys.

The priority system operates as follows: Each consultant's priority value within a project is listed in a specific location within the random vector. For any given project and task, a specific subset of this vector is extracted. The position of a consultant in the availability list directly corresponds to the same position in this priority subset. The priority of each consultant is represented by their respective random value; a lower value signifies a higher priority, thereby increasing the likelihood that the consultant will be assigned to that specific task within the project. However, the assignment can only proceed if the consultant's available hours can accommodate the task's requirements. If the initially prioritized consultant cannot meet these hours, the system moves on to evaluate the next consultant in line based on the priority values. This process continues until either a suitable consultant is assigned or all options in the list are exhausted. If no assignment can be made, the project must be declined.

6.1.4 Skill Division

Skill division is a crucial aspect of this problem. When a project request is submitted, the pool of required skills must be strategically allocated among the needed consultants. This division process is essential for optimal resource utilization. However, it is important to note that for projects requiring only a single consultant ($m_p = 1$), the concept of skill division becomes irrelevant, as all required skills are inherently assigned to that single team member. The encoding of skill division is similar to the previous encodings, however, the random keys are now not used for priority but for ordering. The first step is to extract a list of eligible projects from the feasibility filter. Only the skills of eligible projects that need at least two team members ($m_p \geq 2$) should be taken into account. By not including the skills of $m_p = 1$ projects, the unnecessary information is not stored in the chromosome, therefore the search space is reduced and the convergence ability is improved. For a set of n eligible projects where for each project p, where the skills of the projects are gathered into a list denoted as S, the skill division process is encoded through a vector of $|S|$ random keys. The ordering system on these random keys only has to be considered when confronted with a project that requires at least two consultants, since in a project that requires one consultant all skills are allocated to a single consultant. A more in-depth explanation of the mechanics behind this process is detailed in Section 6.2.

6.1.5 Time Windows

In the context of C2Pa, a project could have a flexible start time window. The size of the interval determines the length of the encoding. In other words, the number of possible start times represent the contribution to the chromosome. To keep the size of the chromosome as small as possible, only the possible start times of the eligible projects are included. To be more concise, for a set of n eligible projects, each having the number of possible start times T_i, the time windows aspect of the C2Pa problem is encoded through a vector of $|T|$ random keys. The ordering system is used to translate the random keys to a project start time. For a project, the index of the lowest random value within that project time window vector determines the time offset to the earliest starting time, e.g., if the second index of the list represents the lowest random value the start time of the project should be extended with one week.

6.2 Decoding

The chromosome resulting from the encoding is made up of the concatenation of the four procedures explained in Sections 6.1.2-6.1.5, thus having a length of $Z = P + |N| + |S| + |T|$. Equation 6.1 gives a visualization of the chromosome. Each chromosome, or solution, in the population will have the same length. Within a generation loop of the algorithms, each solution is submitted to the decoder, which returns the schedule, the objective values, and other variables. In addition to the chromosome as an input variable for the decoder, the decoder has more input variables, the most important variables are: the variables connected to the consultants; net available hours, satisfaction and skill levels, and the variables connected to the project side of the problem; project durations, project start dates, project hours, required skill levels.

$$\left(\underbrace{g_1, .., g_P}_{\text{Projects}}, \underbrace{g_{P+1}, .., g_{P+|N|}}_{\text{Consultants}}, \underbrace{g_{P+|N|+1}, .., g_{P+|N|+|S|}}_{\text{Skills}}, \underbrace{g_{P+|N|+|S|+1}, .., g_{P+|N|+|S|+|T|}}_{\text{Time Windows}}\right) \tag{6.1}$$

The schedule generation scheme incrementally adds a project to the schedule based on the priority value of the projects. If we were to name this procedure, the serial schedule generation scheme (S-SGS) of Kolisch and Hartmann (25) comes closest to this approach. The priority system determines the order in which the projects p should be filled. In Figure 6.1, an example can be found that illustrates the priority decoding of five projects. Project E is initially declined, as there are not enough consultants to assign to the project given the requirements. Therefore, the order in which the projects will be filled is:
$B \rightarrow D \rightarrow A \rightarrow C$.

Prior to assigning the consultants to projects, the start dates of the projects have to be decoded from the chromosome. This information is stored in the last $|T|$ random keys of the chromosome. For every project, the corresponding possible offsets from the start time are extracted from the chromosome and the offset with the lowest random key is chosen. Figure 6.2 shows how this approach works. The figure describes the situation where, for example project B, has a flexible time window of 4 weeks, thus having 5 different start times. The next step is to sort the random keys and from there selecting the lowest value. This results in Offset 1 having the lowest value, therefore the start time of project B is extended with one week.

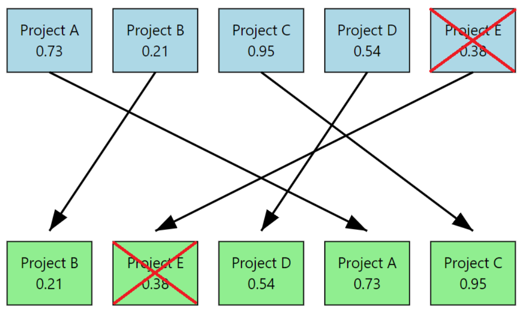

Figure 6.1: Project priority decoding

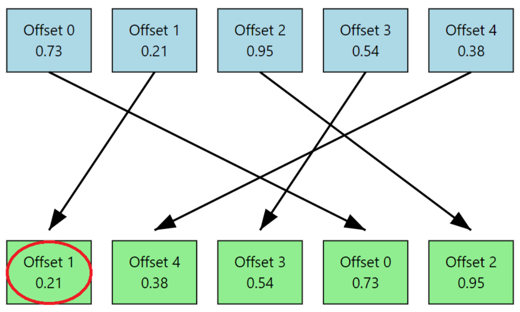

Figure 6.2: Start time decoding

The next step is to assign the consultants to the projects, as the order has been determined. For each project and task, the corresponding subset of the random key vector has to be extracted from the chromosome. These priority values are then sorted to determine which consultant should be assigned to the project. Figure 6.3 illustrates the assignment procedure for two scenarios. The first scenario describes the situation where two projects, for example B and D, with identical start time and utilization of 100% must be scheduled. Project B is indicated with the color blue and project D with color green. From the priority sorting, it can be concluded that consultant 4 is assigned to project B. However, this assignment causes consultant 4 to be unavailable for the other project; therefore, there is no other option to assign consultant 2 to project D. This is, of course, all within the bounds of their working hours. In the second scenario, project C that needs 2 consultants must be scheduled. As earlier, consultant 4 is assigned to task 1. Despite the high priority of consultant 4 for task 2, it is prohibited to assign consultant 4 to task 2 as it is not allowed to be assigned to two tasks within the same project. Consequently, consultant 2 is assigned to task 2 of project C. If consultant 2 is not available for task 2, due to assignment in previous projects, and the availability list for this task is exhausted, the only logical consequence is to decline the project.

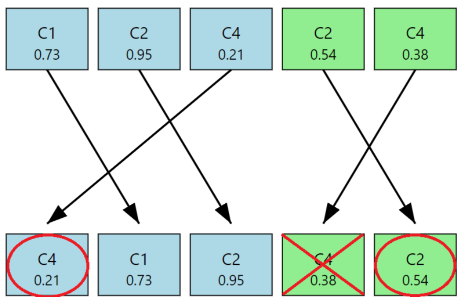

Figure 6.3: Consultant decoding

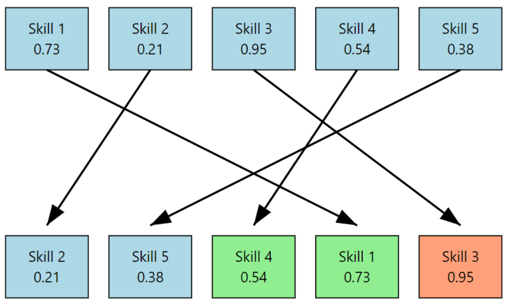

Figure 6.4: Skills decoding

The final step is to assign the required skills to consultants. The decoding procedure of this step is illustrated in Figure 6.4. As explained in Section 6.1.4, this procedure has relevance only when faced with a project that has at least two tasks. The skills of projects that need one consultant are not encoded in the chromosome, these can directly be assigned to the chosen consultant. The skill division decoding for the eligible projects can begin once the right subset of the chromosome has been extracted that corresponds to the required skills of project p. This involves sorting the random keys and splitting the resulting vector in the places that cause each task to have the prescribed number of skills.

The procedure for determining the prescribed number of skills has been described in Section 4.3. Translating this to Figure 6.4, a project that requires five skills to be divided over three consultants in a split of [2, 2, 1] skills results in task 1 having skills 3 and 1, task 2 having skills 2 and 5 and task 3 having skill 4.

Once these steps are complete and the information is stored in the five-dimensional binary variable x, the KPIs can be calculated using this variable. In Algorithm 1, the simplified pseudocode of the decoding procedure can be found. The algorithm includes the four decoding steps in the right order, the feasibility checks, and the condition to update the decision variable x. After all projects have been completed, the KPIs and the number of declined projects are calculated and returned. These are directly used to determine the fitness of an individual.

Algorithm 1 Decoder function

```
1: initialize $x$
2: initialize $ap$
3: translate start date from chromosome
4: translate project order from chromosome
5: for project $p$ in project order do
6:     project_feasbility = True
7:     for each activity $m$ in project $p$ do
8:         retrieve activity hours $ph_{p,m}$
9:         nh, assigned, assigned_cons = assign_consultant_activity()
10:        if assigned == $\emptyset$ then
11:            project_feasbility = False
12:            $ap[p] = 0$
13:        end if
14:    end for
15:    if project_feasibility == True then
16:        assigned_skills = assign_skills_to_consultant()
17:        update $x$ based on assigned consultants and assigned_skills
18:    end if
19: end for
20: calculate KPIs from $x$
21: calculate number of declined projects from $ap$
22: return KPIs, number declined projects
```

6.3 Objective Function

As described in Chapter 5, the objective function or fitness function consists of five components; the satisfaction KPI, the skill match KPI, the hourly cost KPI, the utilization rate KPI, and the number of declined projects. In Sections 6.3.1-6.3.4, an explanation will be given on how these KPIs are calculated in our non-exact algorithm. Section 6.3.5 will explain how the KPIs are weighted and in Section 6.3.6 will present the complete objective or fitness function.

6.3.1 Satisfaction KPI

Satisfied consultants are important for many reasons, for example, satisfied consulants are more likely to remain with the organization, be motivated and engaged in their work, and provide quality service to clients. This can ultimately contribute to a positive work environment and increase team morale. The satisfaction KPI is calculated as the average satisfaction of the consultants assigned to projects given the required skills. This calculation is done over the entire duration of accepted projects over all required skills and all team members. This sum is then divided by the sum of the product of the total required skills and the total duration of the accepted projects. The calculation can be found in Equation 6.2. The variable x is the binary decision matrix that stores information about the assignments, this variable equates to $x_{i,p,t,m,s} = 1$ when consultant i is assigned to skill s of project p at time t. The variable css stores the consultant satisfaction scores, and variable rs is a indicator matrix that is 1 if skill s is included in project p.

$$\text{Satisfaction KPI} = \frac{\sum_{i=1}^{I} \sum_{p=1}^{P} \sum_{t=ES_p}^{LF_p} \sum_{m=1}^{m_p} \sum_{s=1}^{S} x_{i,p,t,m,s} \cdot css_{s,i}}{\sum_{s=1}^{S} \sum_{p=1}^{P} rs_{s,p} \cdot pt_p \cdot ap_p} \tag{6.2}$$

This equation returns a value within the interval of 1 and 10. The design of the objective function requires that each KPI is scaled to a value that is within the limits of [0, 1]. To achieve that, the resulting average satisfaction score is divided by 10.

6.3.2 Skill Match KPI

The quality of an assignment of a consultant to a project is evaluated by comparing consultants' skills to the project requirements. The skill match score is calculated as the consultant's skill level minus the project's required level. Higher scores suggest better potential quality, but consistently overassigning skilled consultants risks resource misallocation, potentially leaving more complex projects understaffed.

The skill match KPI is calculated as the average skill match per time unit across all required skills for a project. Since consultants can't swap skills during a project, this calculation could be done for any single time unit within the project's duration. The specific time unit chosen doesn't affect the result. Equation 6.3 calculates the skill match KPI. The equation for skill match is divided into two parts. The upper part calculates the negative deviation from the required skill level, which represents underqualification, where the lower part calculates the positive deviation from the required skill level, which represents the overqualification. The deviation is calculated by subtracting the required skill level ($psl_{s,p}$) from the skill level of the assigned consultant on that same skill ($csl_{s,i}$). Underqualification occurs when the skill level of the assigned consultant is lower than the required skill level ($csl_{s,i} < psl_{s,p}$). A perfect match occurs when the skill level of a consultant is identical to the required skill level of the skill of a project, in that case the consultant is qualified. Lastly, a consultant could be overqualified for a skill in a project, or in other words, $csl_{s,i} > psl_{s,p}$. The variables q and w function as indicator variables that enable one to make a distinction between underqualification and overqualification. Thus, the variable $q_{p,s,i}$ is 1 when a consultant is underqualified for a skill of a project, for $w_{p,s,i}$ the inverse is true.

$$\begin{aligned}\text{Skill Match KPI} = &\frac{\sum_{i=1}^{I}\sum_{p=1}^{P}\sum_{m=1}^{m_p}\sum_{s=1}^{S}\left(csl_{s,i}^2 - psl_{s,p}^2\right)\cdot q_{p,s,i}\cdot x_{i,p,t,m,s}}{\sum_{p=1}^{P} ap_p}\\ &+ \\ &\frac{\sum_{i=1}^{I}\sum_{p=1}^{P}\sum_{m=1}^{m_p}\sum_{s=1}^{S}\left(csl_{s,i} - psl_{s,p}\right)\cdot w_{p,s,i}\cdot x_{i,p,t,m,s}}{\sum_{p=1}^{P} ap_p}\end{aligned} \tag{6.3}$$

In the C2Pa problem, the aim is to differentiate between consultants who are perfectly qualified and those who are mismatched in their skills. Initially, a consultant with equal overqualification and underqualification across different skills would yield the same skill match score as a perfectly qualified consultant, which is undesirable. To address this, it has been decided to penalize underqualification more heavily by squaring the skill levels before comparison. This approach has two benefits: it more severely penalizes larger skill gaps, which could indicate significant mismatches, and it prevents overqualification in one area from fully compensating for underqualification in another. The result is a more balanced evaluation that better reflects the preference for consultants whose skills align closely with the project requirements.

The following approach is used to reduce the average skill match per project to a value within the interval of [-1, 1]. For each project, the maximum required skill level is extracted, then squared, and lastly multiplied by the number of skills of the project. These values are summed and act as the normalizing constant for the skill match KPI. In this way, the yielded KPI value that will be used in the objective function represents the relative skill match performance compared to the scenario where each skill is mismatched. In addition to this, the constant dynamically adjusts itself when faced with different instance sizes.

Lastly, to make the resulting skill match value more intuitive in the evaluation section, we have decided to use a different notation of skill match in the model and in the results. The model will utilize the described skill match above, whereas the skill match returned to the user will have the following modifications: first, the skill match will be multiplied by its normalizing constant. The resulting skill match will be divided by 9 if it is negative, whereas it will be divided by 2 in case it is positive. Division by 9 corresponds to having a complete underqualification mismatch (as the worst scenario is $3^2 - 0^2 = 9$), where division by 2 corresponds to having a complete overqualification (as the worst scenario is $3 - 1 = 2$). The definition of skill match that the user will be confronted with is the average mismatched skills per project, where a negative value implies underqualification and a positive value overqualification.

6.3.3 Hourly Cost KPI

The KPI for hourly cost of assigned consultants calculates the average cost per hour of assignment. The calculation of this KPI can be found in Equation 6.4. It considers a consultant's hourly rate (c_i) when they're assigned to project skills ($x_{i,p,t,m,s} = 1$). The KPI is computed by summing the costs of all assignments and dividing by the total hours of the skills of accepted projects. This results in a weighted average hourly cost for the consultants assigned to projects. The normalizing constant is set to the maximum hourly rate of a consultant, which is set to 220.

$$\text{Hourly Cost KPI} = \frac{\sum_{i=1}^{I} \sum_{p=1}^{P} \sum_{t=ES_p}^{LF_p} \sum_{m=1}^{m_p} \sum_{s=1}^{S} c_i \cdot x_{i,p,t,m,s}}{\sum_{s=1}^{S} \sum_{p=1}^{P} rs_{s,p} \cdot pt_p \cdot ap_p} \tag{6.4}$$

Incorporating this KPI into the objective function enables us to introduce a bias for consultants of lower job positions, as lower job positions are usually favoured more to gain more experience. However, the impact of this bias should be limited as it would negatively affect the other KPIs.

6.3.4 Utilization Rate KPI

The KPI consultant-to-project utilization rate has undergone an overhaul in terms of calculation compared to (1). Originally, the KPI is calculated as the average utilization per consultant, which is the sum over all chargeable project hours of the consultant divided by the product of the sum over all working hours of the consultant ($WH_{i,t}$) and time horizon (T). This method may be effective for small departments, but it becomes problematic when applied to an entire practice. The larger scale introduces flaws that the approach does not address. The primary flaw arises from the mismatch between project scale and consultant numbers when considering an entire practice rather than a single department. With a large consultant pool and relatively few projects, accepting a small number of projects has minimal impact on the average consultant-to-project utilization rate. This rate should also account for both client and internal projects. Consequently, other KPIs have a more significant influence on the objective value, effectively diminishing the utilization rate's contribution to the objective value.

To address this flaw, a modification to the calculation is proposed, which can be seen in Equation 6.5. In pursuit to make the KPI more impactful, it has been decided to only use the information about the projects that are subject to be scheduled. Secondly, the information regarding individual working hours is redundant as the combination of a client project variable $(1 - beach_p)$ and the accepted projects variable (ap_p) also indicate which projects contributed to the client project hours. The source of the hours, whether from specific consultants or skills, is irrelevant. The crucial factor is the total number of hours dedicated to client projects. Consequently, the KPI is calculated as the sum of the total hours over all accepted client projects divided by the total hours of client, internal and rejected projects. No normalization constant is needed for this KPI, as it is already scaled to fit in the interval of [0, 1]. This formulation allows for noticeable fluctuations in KPI value when faced with decisions about which projects to decline. Practice utilization is a key performance metric for the company, and as such, it will be displayed to the end-user. However, in the optimization process, the model uses the modified utilization in its fitness function defined by Equation 6.5.

$$\text{Utilization Rate KPI} = \frac{\sum_{p=1}^{P} \sum_{m=1}^{m_p} (1 - beach_p) \cdot ap_p \cdot pt_p \cdot ph_{p,m}}{\sum_{p=1}^{P} \sum_{m=1}^{m_p} pt_p \cdot ph_{p,m}} \tag{6.5}$$

6.3.5 Weighting the KPIs

In multi-objective optimization, the assignment of weights to different KPIs is a crucial step that significantly influences the outcome. This analysis focuses on two prominent methods for criteria weighting: the Rank Order Centroid (ROC) method and the Point Allocation method.

The Rank Order Centroid method, introduced in (26), is a technique that converts ordinal rank information into numerical weights. It is based on the centroid of the weight space consistent with the provided rank order. For n criteria, the ROC weights are given by:

$$\text{weights} = \left[\frac{\sum_{i=k}^{n} \frac{1}{i}}{n} \mid k \in \{1, 2, \ldots, n\} \right] \tag{6.6}$$

The advantages of ROC can be summarized as follows, the method is simple in implementation and use, it only requires ordinal ranking of criteria, it provides consistent results for a given ranking, and it minimizes cognitive load on decision-makers. On the other hand, there are also challenges associated with ROC, it assumes a specific pattern of decreasing importance, it may overemphasize top-ranked criteria, it has limited flexibility in capturing nuanced preferences, and it is unable to incorporate preference information.

The Point Allocation method involves the direct assignment of numerical values to criteria based on their perceived importance (27). These numerical values are weighted to produce the individual weights. For n criteria with assigned points p_i, the weight of the i^{th} criterion w_i is calculated by:

$$w_i = \frac{p_i}{\sum_{j=1}^{n} p_j} \tag{6.7}$$

The pros of Point Allocation include that it is intuitive and straightforward for stakeholders to understand, it allows for nuanced expression of preferences, it is flexible and easily adjustable based on context, it can capture non-linear differences in importance, and it is a direct representation of preferences. While the benefits are clear, it is also important to consider the potential drawbacks, it is more subjective, potentially leading to inconsistencies. Moreover, it requires careful consideration in point assignment.

Although both methods aim to derive numerical weights from preference information, they differ significantly in their approach and outcomes. ROC requires only a ranking, making it simpler to implement. Point Allocation demands more cognitive effort in assigning numerical values. Point Allocation offers greater flexibility, allowing for fine-tuning of weights. ROC weights are fixed once the ranking is determined. Point Allocation can capture more nuanced differences in importance. ROC assumes a specific pattern of decreasing importance that may not always reflect reality.

In order to align the weights of the KPIs, and therefore the behavior of the models, with the prioritization and strategic objectives of the OTL team, an interview with the OTL team was conducted. The OTL team emphasized that the primary focus should be on skill match, as this is crucial for ensuring the quality and success of the project. By aligning the consultant's expertise with the project requirements, the team can secure optimal outcomes. Following this, utilization is the next priority. Consultant satisfaction was identified as the third key KPI, recognizing that a motivated and content consultant contributes positively to project performance. The hourly cost should be considered at last, but only as a tiebreaker when consultants' skills and/or satisfaction are comparable.

In such cases, preference would go to lower-level consultants to maximize the utilization rate of newly joined consultants.

For the C2Pa problem, the Point Allocation method appears to be more suitable for several reasons, it allows for precise expression of the relative importance of skill match compared to other factors, it can easily accommodate the use of cost as a tiebreaker by assigning it a low but nonzero weight, it provides the flexibility to adjust weights based on specific project requirements or changing strategic objectives, and it can capture nonlinear importance differences between criteria, which may be more reflective of real-world priorities in consultant assignment. Although ROC offers simplicity and consistency, the additional flexibility and nuance provided by the Point Allocation method align better with the complex nature of the C2Pa problem.

The information gathered from the interview on the prioritization of KPIs has been translated into numerical values ranging from 1 to 10 as follows: Since skill match is by far the most significant KPI, it is assigned a value of 10. To ensure that client projects are prioritized over internal projects, utilization is assigned a value of 7. The satisfaction KPI, while less important, is still relevant and is assigned a value of 4. Although this value does not significantly impact the model's solution characteristics, it moderately incorporates the consultants' preferences. Finally, to encourage the selection of lower-level consultants over more experienced ones when qualifications are similar, and to moderately favor lower levels in the assignment process, hourly cost is assigned a value of 2. Incorporating the assignment of [10, 7, 4, 2] to the weights of skill match, utilization, satisfaction, and hourly cost in the Point Allocation method results in weights of [0.4348, 0.3043, 0.1739, 0.0869].

6.3.6 Fitness Function

The fitness function is a critical component of evolutionary algorithms, guiding the selection and evolution of candidates over successive generations. It quantifies how well a given candidate solution scores compared to the population, effectively steering the evolution process towards optimal solutions. One of the most frequently used techniques to convert multiple objectives into a single objective is the weighted sum method. Given the wishes of the OTL team, as indicated in Section 6.3.5, scalarization of the KPIs is a natural result. The relative importance of the KPIs has already been established. Utilizing these weights in the weighted sum method combines the four KPIs and turns it into a singular value. This value can be used in the evolution process.

As mentioned earlier, the fitness function of C2Pa consists of five components; number of declined projects, skill match KPI, utilization rate KPI, satisfaction KPI, and hourly cost KPI. Equation 6.8 presents the fitness function. The equation incorporates the weights, the normalizing constants and the KPI values. The minimization optimization objective resulted in the following behavior of the KPIs: The ambition is to have perfect skill match, any deviation from 0 needs to be punished, therefore, the absolute value of this KPI needs to be added. The utilization on client projects needs to be as high as possible, thus the utilization needs to be subtracted. For satisfaction, the same holds. Consultants of lower position need to be assigned in favor of higher-positioned consultants, therefore, this KPI needs to be added. Moreover, declining a project yields a penalty of 2. Consequently, a solution that declines a project has no chance to compete with non-declining solutions.

$$\begin{aligned}\text{Fitness} =& 2 \cdot \#declined + \left| \frac{0.4348 \cdot Skill\_Match}{norm\_skill\_match} \right| - 0.3043 \cdot utilization \\ & - \frac{0.1739 \cdot satisfaction}{10} + \frac{0.0869 \cdot cost}{220}\end{aligned} \tag{6.8}$$

An important aspect of evolutionary algorithms is the computational efficiency of the fitness function. In one evaluation of the fitness function, four separate heavy calculations are required. Computing these with plain Python is ridiculously slow. These problems have been significantly reduced in four ways: [1] only use relevant subsets of the variables [2] use vectorization of computations [3] extensive use of NumPy [4] transforming the computations into Numba (28) compatible code.

6.4 Local Search

As concluded in Chapter 3, intensification methods can be crucial to achieve high performance in complex optimization problems. Local search is an intensification method that focuses on exploring the immediate neighborhood of a current solution to find improvements. When combined with an evolutionary algorithm, this approach creates a hybrid method that leverages the global exploration capabilities of EAs with the local exploitation strength of local search techniques. Although local search can improve solution quality, it often comes at the cost of computational efficiency. This intensification method typically examines all options in the immediate neighborhood of a solution, also called neighborhood moves, which can be time-consuming, especially for problems with large solution spaces. Given that computing the fitness value is the main bottleneck for the efficiency of the algorithms, the number of individuals on and frequency of executing local search should be carefully determined.

In this research, two types of local search have been developed, *Consultant Swap* and *Skill Swap*. These will be discussed in Section 6.4.1 and Section 6.4.2, respectively.

6.4.1 Consultant Swap

Consultant Swap local search entails the neighborhood move of replacing an assigned consultant of a project with a consultant that has not been assigned to that project. Swapping a consultant impacts three KPIs: the satisfaction KPI, the skill match KPI, and the hourly cost KPI. Whenever a consultant is subject to a swap, these KPIs need to be reevaluated to judge whether an improvement has been made.

The consultant swap procedure is presented in Algorithm 2. A predefined duration of three iterations has been allocated to this procedure. The first step in an iteration is to shuffle the project ordering. After that, the following iterative process will be executed. For every activity of an accepted project, the assigned consultant will be retrieved together with the consultants assigned to the other activities. From there, the unassigned but available consultants will be retrieved. In order to assess whether these unassigned consultants are a better match for the activity of a project, the indices of both the assigned and unassigned consultant in the chromosome must be retrieved. After retrieving the indices, the random keys of these consultants are swapped in the chromosome. This swap now causes the unassigned consultant to be assigned. The modified chromosome is fed to the decoder, which returns the fitness. If the resulting fitness is lower than the original fitness, then it will be stored in an improvement list. After the available consultants have been exhausted, the highest improvement will be selected to be implemented in the chromosome. This greedy approach is chosen because the number of consultants available can be quite large. That, combined with the fact that local search will only be applied to solutions of the elite, thus the solution quality is high, justifies the decision to implement a greedy approach.

Algorithm 2 Consultant Swap local search

```
1: Input: $x$, chromosome, net_hours
2: Output: $x$, chromosome, fitness
3: best_fitness ← -satisfaction + |skill match| + cost
4: for i in range(iterations) do
5:     shuffle project order
6:     for project $p$ in projects do
7:         for activity $m$ in roles[$p$] do
8:             best_improvement ← $\emptyset$
9:             retrieve available consultants $ac_{p,m}$ of activity $m$ of project $p$
10:            retrieve assigned consultant $c_1$ of activity $m$ of project $p$
11:            for consultant $c_2$ in $ac_{p,m}$ do
12:                swap consultants $c_1$ and $c_2$ in chromosome
13:                submit chromosome to decoder → retrieve temp_fitness
14:                if temp_fitness < best_fitness then
15:                    best_fitness ← temp_fitness
16:                    update best_improvement to this swap
17:                end if
18:            end for
19:            if best_improvement $\neq \emptyset$ then
20:                implement best improvement in chromosome
21:            end if
22:        end for
23:    end for
24: end for
25: $x$, fitness ← decoder(chromosome)
26: return $x$, chromosome, fitness
```

6.4.2 Skill Swap

Skill Swap local search entails the neighborhood move of swapping one skill between two team members of a project. The number of skills that each consultant needs to fill remains the same, however, one skill of each consultant is interchanged with a skill that was earlier assigned to the other consultant. Consequently, such a swap impacts only two KPIs: the satisfaction KPI and the skill match KPI. Therefore, only these KPIs will be taken into account when performing swaps.

The skill swap procedure is presented in Algorithm 3. Initially, the fitness of the individual has to be calculated, as this will be used to check for improvements. Next, a subset is created that only contains the projects that need multiple consultants. Each project will then enter a closed loop in which swaps will be executed. Within a project, every possible inter-consultant skill swap will be evaluated. A swap is executed by swapping the random keys of two consultants. In order to do this correctly, the index of these skills in the chromosome needs to be extracted. After a swap, the altered chromosome is submitted to the decoder, which returns the KPIs that are needed for the calculation. The information of the swap will be stored if it leads to an improvement. When all swaps in a project have been executed, an improvement will be chosen by means of probability. Probabilities are calculated using positive improvement as a weighting factor. The improvement list will be updated to no longer include the involved swapped skills. From there, only the unknown swaps that are created by the earlier swap have to be evaluated. For every multi-consultant project, this procedure will be carried out until the improvement list is exhausted. Choosing an improvement based on probabilities made this approach less greedy compared to choosing the biggest improvement. This is favourable since the search space of dividing skills over multiple consultants is more complex compared to assigning consultants to an activity.

Algorithm 3 Skill Swap Local Search

```
1: Input: $x$, chromosome
2: Output: $x$, chromosome, fitness
3: best_fitness $\leftarrow$ -satisfaction + |skill match|
4: for project in multi consultant projects do
5:     improvements $\leftarrow$ [ ]
6:     for activity $m_1$ in roles[project] do
7:         for activity $m_2$ in roles[project], $m_2 \neq m_1$ do
8:             for skill $s_1$ in $m_1$ do
9:                 for skill $s_2$ in $m_2$ do
10:                    swap skills $s_1$ and $s_2$ in chromosome
11:                    submit chromosome to decoder $\rightarrow$ retrieve temp_fitness
12:                    if temp_fitness < best_fitness then
13:                        append information swap to improvements
14:                    end if
15:                end for
16:            end for
17:        end for
18:    end for
19:    if improvements $\neq \emptyset$ then
20:        sample and implement improvement in chromosome
21:        delete improvements that involve chosen skills from list
22:        update best_fitness
23:        if improvements $\neq \emptyset$ then
24:            while improvements $\neq \emptyset$ do
25:                repeat lines [6:22], where $s_1$ is replaced with both swapped skills
26:                lines 6 & 8 are neglected, line 7 is modified
27:            end while
28:        end if
29:    end if
30: end for
31: $x$, fitness $\leftarrow$ decoder(chromosome)
32: return $x$, chromosome, fitness
```

6.5 BRKGA

This section details the methodology of the developed BRKGA. We begin by describing the core components of the algorithm, including the variation and selection operators. Following this, the hyperparameter tuning approach will be explained.

6.5.1 Methodology

The Biased Random-Key Genetic Algorithm (BRKGA) is a metaheuristic optimization method within the broader family of genetic algorithms. This approach was first introduced by Gonçalves and Resende (24), building upon and enhancing the foundations laid by Bean's Random-Key Genetic Algorithm (23). Genetic Algorithms are search heuristics that mimic the process of natural evolution. They operate on a population of potential solutions, applying the principles of selection, crossover, and mutation. In these systems, *survival of the fittest* is the core principle, which means that solutions with higher fitness are more likely to survive and reproduce. Over time, this process can lead to the evolution of increasingly better solutions.

BRKGA starts with the generation of an initial population. As mentioned in Section 6.2, a chromosome consists of random keys of size Z. Once the population of size λ has been randomly generated, the population has to go through the decoding function. This function returns the values of the KPIs, the information on the acceptance of projects, and the decision variable x. The values of the KPIs and the number of declined projects are then used to calculate the fitness value, as indicated in Equation 6.8.

The evolutionary strategy is where the name *biased* is introduced. Reproduction and crossover operators select which individuals will produce offspring and dictate how genetic information is shared between parents to generate new individuals. These processes typically improve the overall quality of populations and promote convergence. In contrast, mutation counteracts this trend by allowing random changes in genetic material, thus introducing diversity into the population. In the reproduction procedure, some of the best individuals of the current generation are directly copied to next generation. This approach is also called *elitist* and its advantage is that it promotes improving quality solutions, moreover, it can also lead to fast convergence. The percentage of the best population that is directly copied to the next generation is called elite rate p_e. This splits the population into two groups, the elite and the non-elite.

Regarding the crossover operator, two-point crossover is used. Two-point crossover is chosen because it is able to preserve blocks of successful assignments better than one-point crossover, and is less disruptive compared to uniform crossover. It is favorable to keep blocks of the chromosomes intact, as the reproduction procedure is biased. In crossover, two solutions are randomly chosen to act as parents. One parent is from the elite individuals, the other is from the non-elite individuals. With that, BRKGA applies a selection process that shows a preference for elite solutions. This *bias* accelerates the convergence toward high-quality solutions by ensuring that genetic material from the best performing individuals is more likely to be passed on to subsequent generations. In a two-point crossover function, two random points, p_1 and p_2, are selected along the length of the parent chromosomes, where $p_1 < p_2$. The blocks of genes between these points are swapped between two parent chromosomes to create two new offspring children. Figure 6.5 shows how two-point crossover generates the offspring. Subsequently, mutation operators are applied to the offspring where each gene has a mutation probability of p_m. This introduces diversification into the population. In addition to this, the new offspring consisting of the elite and the generated children is supplemented with randomly generated individuals, who are responsible for p_r of the population. This means that in total $\lambda - p_e - p_r$ children are generated by the crossover procedure. A visual summary of the reproduction procedure is illustrated in Figure 6.6 (adapted from (15)).

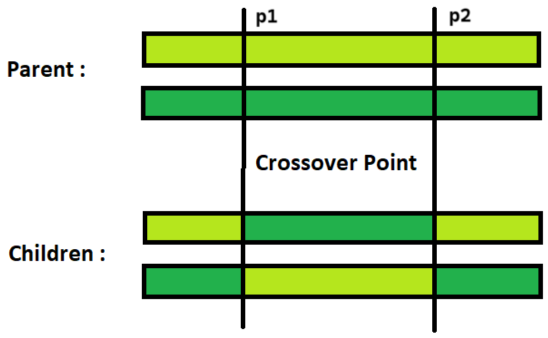

Figure 6.5: Two-point crossover

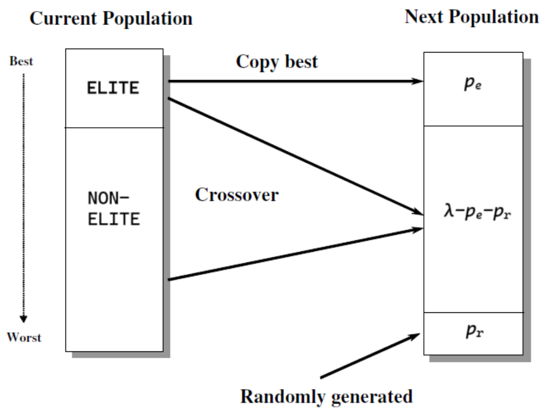

Figure 6.6: Reproduction BRKGA

Additional measures are implemented to prevent early convergence and to direct the evolutionary process. Firstly, the idea proposed by Chaves et al. (29) is implemented, which makes the parameters p_e and p_r adaptive. The values of the parameters will range within a predefined interval in which p_e increases and p_r decreases over the generations. By increasing elitism and reducing random replacement, the algorithm can dynamically shift focus. Early generations emphasize diversity and discovery of promising solutions. Later generations focus on refining those solutions, maximizing their quality. Secondly, the *shaking* operator of Andrade et al. (30) is implemented. The shaking operator is used to maintain diversity in the non-elite set and preserve the convergence structure in the elite set of the population. It helps in introducing random modifications to elite individuals while resetting non-elite individuals. The shaking process ensures that the population does not get stuck in local optima and explores different solutions effectively.

The shaking operator can be activated in three ways:

- the fitness of the entire elite is identical
- the best fitness score has not been improved for R generations
- the fitness score has not been improved for R^* generations.

During the shaking process, elite individuals undergo perturbations based on the shaking intensity parameter Ψ, which is proportional to the length of the chromosome. Perturbations include swapping of random keys and replacing keys with random numbers. For each shaking scenario, the shaking intensity Ψ is uniformly sampled from a scenario dependent interval. When the fitness of the elite is identical, Ψ is sampled from the interval [0.05, 0.30], when the best solution has not improved for R generations, Ψ is sampled from [0.20, 1.0], and lastly, when the best solution has not improved for R^* generations, Ψ is sampled from [0.40, 1.0]. Thus, different scenarios result in a more disruptive shaking sequence.

Local search will be executed a total of 10 times in one run, which is the base frequency for the base instance of three projects. Executing local search too frequently will significantly decrease the efficiency of the models, while infrequent usage of local search will not have much impact. It will be called right after the reproduction procedure as the new population is generated. The method used to select the individuals that will undergo local search is called Tournament Selection. Tournament selection is widely recognized as an efficient method for selecting individuals for local search in evolutionary algorithms, particularly due to its adjustable selection pressure and computational efficiency (Ishibuchi et al.(31)). The method works by randomly selecting k individuals from the population and choosing the fittest among them, which helps maintain a balance between diversification and intensification. Unlike random selection, tournament selection ensures higher quality individuals have a better chance of being selected while still maintaining population diversity, as even less fit individuals have a chance of winning smaller tournaments. For BRKGA, the best 75% of the population is available to be selected. To maintain diversity and ensure good selection pressure the tournament size has been set to $k = 3$.

Initially, Consultant Swap Local Search will be performed on five individuals chosen by five iterations of Tournament Selection. Duplicate tournament winners are not allowed. The decision to apply it more frequently but only on a select number of individuals results in faster improvements in early generations, while the diversity of the population is preserved. Afterwards, Skill Swap Local Search will be executed on these same individuals. Following completion of each BRKGA iteration, a final local search phase is implemented. During this concluding stage, the algorithm selects five individuals at random from the top-performing 20% of the population for local search optimization. Post-generation local search can significantly improve the quality of final solutions by fine-tuning the best individuals.

In Figure 6.7, a schematic overview of BRKGA is presented. It includes the elements discussed above. The model stops when a predefined number of generations have been reached. In Chapter 7, the difference in performance and efficiency of a BRKGA model with and without local search will be investigated.

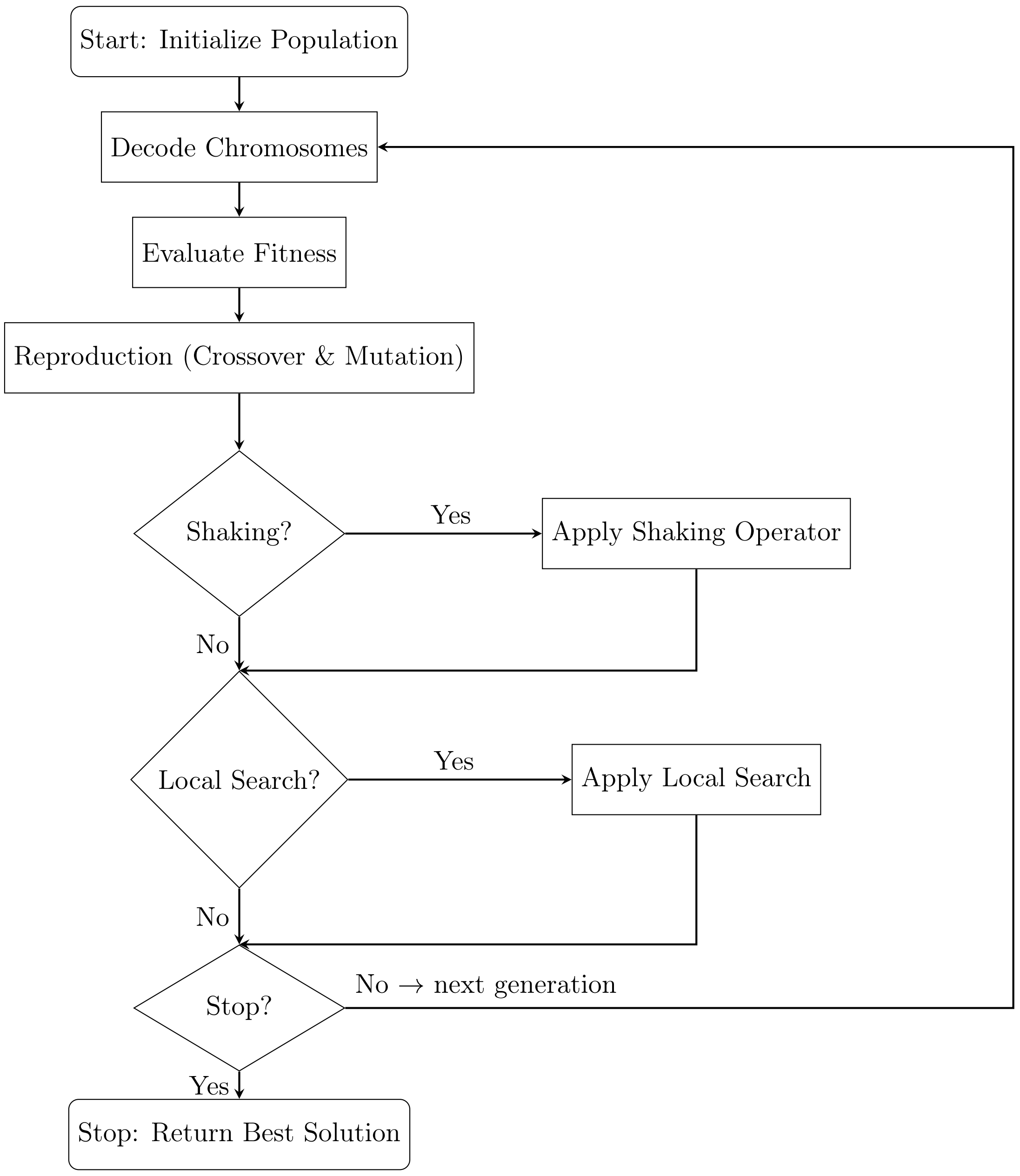

Figure 6.7: Schematic overview BRKGA

6.5.2 Hyperparameter Tuning

Hyperparameter tuning is crucial in optimizing the performance of evolutionary algorithms. Hyperparameters are the configurable settings that control an algorithm's behavior, such as population size, mutation rate, or elite rate, which are not learned from the data but set prior to the learning process. Its importance lies in improving convergence speed, solution quality, robustness, and computational efficiency. As Eiben and Smit (32) emphasize, an evolutionary algorithm with good parameter values can be orders of magnitude better than one with poorly chosen parameter values, and finding optimal values is time-consuming but essential. In this research, the Taguchi method (33) is used to setup the configuration of the experiments.

For BRKGA, the set of hyperparameters that needs tuning is listed in the first column of Table 6.1. The parameters related to the shaking procedure, R and R^*, are not included in the list, these have been fixed to values of 0.25×generations and $R\times1.5$ respectively. The values of these variables are determined based on values used in (30). Table 6.1 presents the set of possible values for population size, mutation rate, elite rate, and respawn rate. These values are based on commonly used settings in the literature. The experimental setup for the Taguchi tuning procedure consists of 9 hyperparameter configurations. Each configuration is executed for 10 independent runs, with each run lasting 170 generations. This shortened duration, compared to a complete model run, allows for investigating the algorithm's early convergence ability and overall performance. Furthermore, local search has not been enabled while tuning since this allows us to establish a baseline performance for the algorithm using just the evolutionary components. This baseline helps to isolate the specific impact of adding local search later. In this way, local search does not overshadow poor baseline parameter choices.

parameter	**values**
pop_size λ	[80, 100, 120]
mutation rate p_m	[0.02, 0.035, 0.05]
elite rate p_e	[[0.10, 0.25], [0.15, 0.30], [0.20, 0.35]]
respawn rate p_r	[[0.05, 0.20], [0.075, 0.225], [0.10,0.25]]

Table 6.1: BRKGA parameter grid

The data used in this experiment is described as follows: the instance consists of four projects with a duration of 10 weeks, 10 weeks, 17 weeks and 16 weeks. These projects are handpicked from the provided data set. This is done to ensure that the test instance is sufficiently complex. Each project requires 1, 2, 2, and 4 consultants. For these projects, the number of skills involved is distributed as 12 skills, 10 skills, 16 skills, and 47 skills. After each run, the metrics covering the KPIs and the objective are stored for further analysis.

The experimental results are summarized in Table 6.2, where an aggregated analysis of the data is presented. This table presents the five best performing configurations sorted by average objective value. The table contains several metrics: average mismatched skills, average hourly cost, and average satisfaction. In particular, utilization has been omitted due to the absence of declined projects, which would have resulted in identical values. Among these metrics, two stand out: the average objective and the average mismatched skills. The average objective serves as a valuable indicator of the model's optimization capabilities, providing insights into its overall performance. Meanwhile, the average mismatched skills is the most important KPI, directly reflecting the quality of assignments.

parameter set	**obj**	**mismatched skills**	**cost**	**satisfaction**
λ: 100, p_m: 0.035, p_e: [0.2, 0.35], p_r: [0.05, 0.2]	**-0.3307**	**-1.5667**	142.1620	**6.2035**
λ: 120, p_m: 0.02, p_e: [0.2, 0.35], p_r: [0.1, 0.25]	-0.3305	-1.7111	141.6662	6.3508
λ: 100, p_m: 0.02, p_e: [0.15, 0.3], p_r:[0.15, 0.3]	-0.3257	-1.8139	**140.4259**	6.1734
λ: 120, p_m: 0.035, p_e: [0.1, 0.25], p_r: [0.15, 0.3]	-0.3248	-1.7500	142.3367	6.0841
λ: 120, p_m: 0.05, p_e: [0.15, 0.3], p_r: [0.05, 0.2]	-0.3247	-1.7500	142.3994	6.0799

Table 6.2: BRKGA tuning top 5 results

From Table 6.2 can be concluded that the hyperparameter set:{ λ: 100, p_m: 0.035, p_e: [0.2, 0.35], p_r: [0.05, 0.2]} produced the best results. In the upcoming experiments in Chapter 7, the BRKGA will use this set of hyperparameters. For each metric, the best result is highlighted in bold. It shows that this configuration outperformed every other configuration on three of the four metrics.

Figure 6.8 presents a detailed overview of the objective value distributions for each hyperparameter set. The boxplot indicates that the selected set achieved the best score out of all sets and the best median score of the sets. Besides that, the boxplot of this selected set does not contain an outlier. It can be concluded that the scores of the best set and the second best set are very close, both in terms of average objective value and the spread of the objective values. However, the best set achieved the scores with a smaller population size, such as $\lambda = 100 < 120$, therefore, the best set will be used as the hyperparameter configuration in the BRKGA.

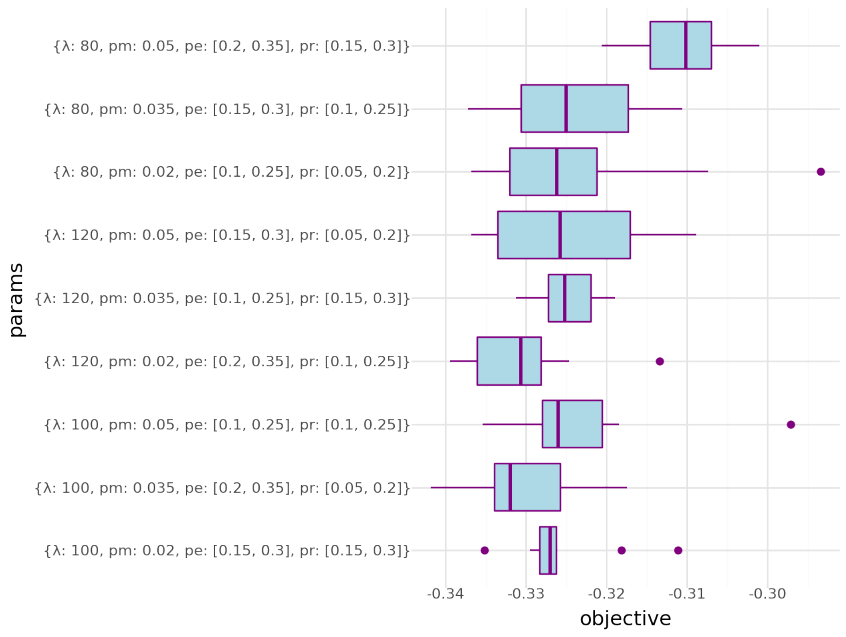

Figure 6.8: Boxplots of objective values of hyperparameter sets BRKGA

6.6 Scatter Search

This section details the methodology of the developed Scatter Search algorithm. The core components of the algorithm will be explained. In addition, the hyperparameter tuning procedure will be summarized.

6.6.1 Methodology

Scatter Search is a population-based metaheuristic, first proposed by Glover (34) in 1977. It can be placed in the category of evolutionary algorithms. However, in contrast to other evolutionary algorithms, such as a genetic algorithm, scatter search is based on the idea that carefully structured techniques for generating new solutions can offer major advantages over methods that rely primarily on randomization. It utilizes strategies for diversification and intensification that have proven to be effective.

The algorithm is compatible with the random key vector representation. The core components of the algorithm are extracted from Marti et al. (35) and Van Peteghem and Vanhoucke (20). (35) outlines the principles of scatter search for both the basic and advanced design. They state that scatter search consists of five methods; a diversification generation method to generate a collection of diverse trial solutions, an improvement method to transform a trial solution into a better solution, a reference set update method to generate and adjust a reference set, a subset generation method that acts on the reference set to produce pairs of solutions that will act as input for the reproduction procedure, and a solution combination method that uses the subset of solutions to produce new trial solutions. In the following sections, these methods and their implementation will be discussed.

The diversification generation method is the first stage of the algorithm. This encompasses the generation of a good and diverse initial population. This method can be performed in sophisticated ways; however, since advanced methods are not applicable to the problem at hand, it has been decided to implement it as follows: initially, a random population of size $4 \cdot \lambda$ will be randomly generated, where population size λ is calculated as: $b \cdot (b_1 + b_2)$. Here, b is a constant and b_1 and b_2 are the sizes of the reference sets. The population will be evaluated and sorted on fitness value. Individuals with duplicate fitness will be removed from the population. Lastly, the λ individuals with the lowest values for the fitness are selected for entrance in the initial population P. Elimination of duplicates ensures diversity.

The improvement method, as it is described in the papers, is not implemented in this algorithm. This is because it is partly used as a repair mechanism, which is unnecessary as every solution is feasible in the decoder and because it is very costly to apply the developed local search to every individual.

After evaluating the population of size λ, two diverse populations are constructed from solution set P, also called reference sets.

- *RefSet* B_1: Contains the b_1 highest-quality solutions from P. To ensure diversity, a minimum distance threshold t_1 is enforced between all the solutions in B_1. If the instance can not provide b_1 solutions for set B_1, the set B_1 is supplemented with randomly chosen solutions from the upper half of set P that are not already included in B_1.

- *RefSet* B_2: Contains b_2 solutions chosen for their diversity. These are selected from the remaining solutions $(P \setminus B_1)$ based on their distance to all solutions in B_1. A larger threshold t_2 (where $t_2 > t_1$) is used to measure this distance, further promoting diversity. The distance comparisons will use solutions in descending fitness order. The best b_2 solutions that are at least at t_2 distance from all solutions in B_1 are allowed to enter B_2. If the number of solutions in B_2 is less than b_2, the set B_2 is supplemented with randomly generated solutions.

The distance between solutions is a measure for diversity, the calculation of the distance between solution s_1 and s_2 can be formulated as follows:

$$\begin{aligned} d_{s_1,s_2} = & \sum_{p=1}^{P} \begin{cases} 0 & \text{if } ap_p^{s_1} = ap_p^{s_2} \\ 1 & \text{otherwise} \end{cases} + \sum_{p=1}^{P} \sum_{m=1}^{m_p} \begin{cases} 0 & \text{if } ac_{p,m}^{s_1} = ac_{p,m}^{s_2} \\ 1 & \text{otherwise} \end{cases} \\ & + \frac{\sum_{p=1}^{P} \begin{cases} 0 & \text{if } o_p^{s_1} = o_p^{s_2} \\ 1 & \text{otherwise} \end{cases}}{5} + \frac{\sum_{p=1}^{P} \begin{cases} 0 & \text{if } t_p^{s_1} = t_p^{s_2} \\ 1 & \text{otherwise} \end{cases}}{4} \end{aligned} \tag{6.9}$$

The calculation consists of four parts, difference in accepted projects covered by the variable ap, difference in accepted consultants covered by variable ac, difference in project ordering covered by variable o, and difference in decoded start date of projects covered by variable t. These will be summed up into one value, which represents the distance between two solutions.

The next phase is to utilize the built reference sets and combine the solutions in a controlled way. This procedure of determining which pairs of solutions are to be combined is called the subset generation method and it functions as follows: First all pairs in B_1 containing at least one new solution compared to the previous generation are combined, leading to two children. The idea of combining two solutions from reference set B_1 stimulates intensification. Next, all pairs of one solution from B_1 and one solution of B_2 are considered. These will also produce two children. This combination is categorized as diversification. The solution combination method specifies how the pairs are processed to generate the children. It has been decided to implement two-point crossover as solution combination method, as (20) concluded that it was superior to other variants in their problem. The combination method is therefore in line with the method used in Section 6.5. In addition, the offspring generated will be exposed to minor mutations at a fixed rate of 3.5%, which ensures that mutations act as small adjustments to explore new possibilities without disrupting the structure of the solution.

Once the offspring has been evaluated, the cycle discussed above repeats itself. Thus, first the reference sets need to be updated. From now on, the solution set P contains the set B_1, set B_2 and the produced offspring. These will first be sorted by fitness, and then the reference sets will be rebuilt. This step is referred to as the reference set update method. In the experimental phase of the research, the algorithm will have a variant in which both Consultant Swap Local Search and Skill Swap Local Search will be applied. On a chosen individual, consultant swap will be executed first. Afterwards, skill swap will be executed on the resulting chromosome of consultant swap. This means that either the chromosome has not changed if no improvements were found, or the altered chromosome from consultant swap will be subjected to skill swap. Since local search is computationally heavy, it will be applied a total of 10 times to 5 individuals of the offspring from the solution combination method. These individuals are selected by means of Tournament Selection with tournament size $k = 4$. The number of iterations of local search is significantly less than what is stated in the literature, where local search is applied to every individual. Following completion of each SS iteration, a local search phase is implemented. For this algorithm, the five individuals are randomly chosen from the combined group of individuals from reference sets B_1 and B_2.

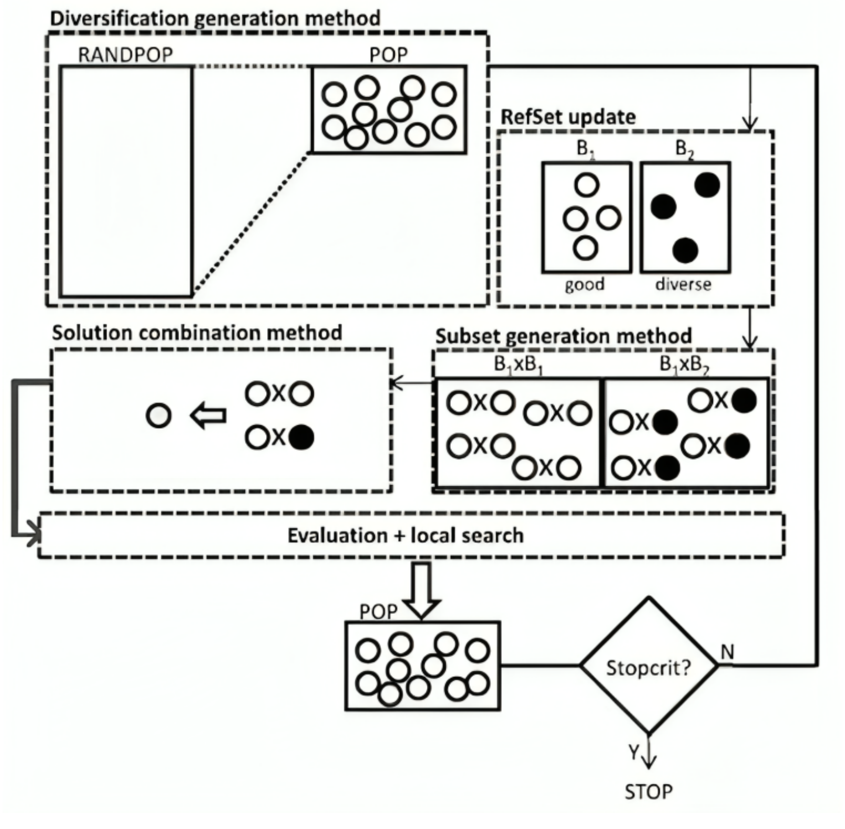

Figure 6.9: Illustrative overview Scatter Search

In Figure 6.9 (adapted from (20)), an overview of the Scatter Search algorithm is presented. The figure includes the loop of the methods that will be executed as long as the stop criterion, which is a predefined number of generations, has not been reached.

6.6.2 Hyperparameter Tuning

The hyperparameter tuning procedure for Scatter Search will be identical to the earlier procedure. That is, generating the experiments with the Taguchi method and storing the results of independent runs for analytical purposes. The hyperparameters of Scatter Search can be found in Table 6.3, where b_1 and b_2 correspond to the size of the reference sets, and the thresholds t_1 and t_2 for the minimal distance between the solutions of reference sets 1 and 2. The thresholds are calculated as: $t_1 \cdot \Sigma_{p=1}^{P} m_p$ and $t_2 \cdot \Sigma_{p=1}^{P} m_p$, which boils down to the threshold times sum of all involved consultants in the projects. Previous tuning experiments indicated that the initial population multiplier b should be equal to 6, as the 5 best performing parameter sets all had $b = 6$.

parameter	**values**
b_1	[6, 8, 10]
b_2	[4, 6, 8]
t_1	[0.2, 0.3, 0.4]
t_2	[0.5, 0.6, 0.7]

Table 6.3: Scatter Search parameter grid

The experimental setup is identical to the previous procedure; thus, the same data instance and performance metrics are used. However, the number of generations has been set to 80, as SS generates more solutions per generation compared to BRKGA. In total, 9 different experiments were conducted.

Table 6.4 presents the results of the five best performing hyperparameter configurations, sorted by average objective value. From the table can be concluded that $\{b_1$: 10, b_2: 8, th_1: 0.3, th_2: 0.5$\}$ is the best performing hyperparameter set. It achieves the highest average score on three of the four performance metrics. Furthermore, if we inspect the boxplots of the objective values of the hyperparameter sets, listed in Figure 6.10, it can be seen that $\{b_1$: 10, b_2: 8, th_1: 0.3, th_2: 0.5$\}$ achieved the best objective value, it has the lowest 1st, 2nd, and 3rd quantile in terms of objective value. In the upcoming experiments in Chapter 7, the Scatter Search algorithm will use this set of hyperparameters.

parameter set	obj	mismatched skills	cost	satisfaction
$\{b_1$: 10, b_2: 8, th_1: 0.3, th_2: 0.5$\}$	**-0.3336**	**-1.5333**	145.0671	**6.3901**
$\{b_1$: 10, b_2: 6, th_1: 0.2, th_2: 0.7$\}$	-0.3293	-1.6167	144.2681	6.2219
$\{b_1$: 8, b_2: 8, th_1: 0.2, th_2: 0.6$\}$	-0.3289	-1.6250	144.4826	6.2170
$\{b_1$: 10, b_2: 4, th_1: 0.4, th_2: 0.6$\}$	-0.3264	-1.8194	**138.9689**	6.1926
$\{b_1$: 8, b_2: 6, th_1: 0.4, th_2: 0.5$\}$	-0.3256	-1.7500	141.4854	6.1128

Table 6.4: Scatter Search tuning top 5 results

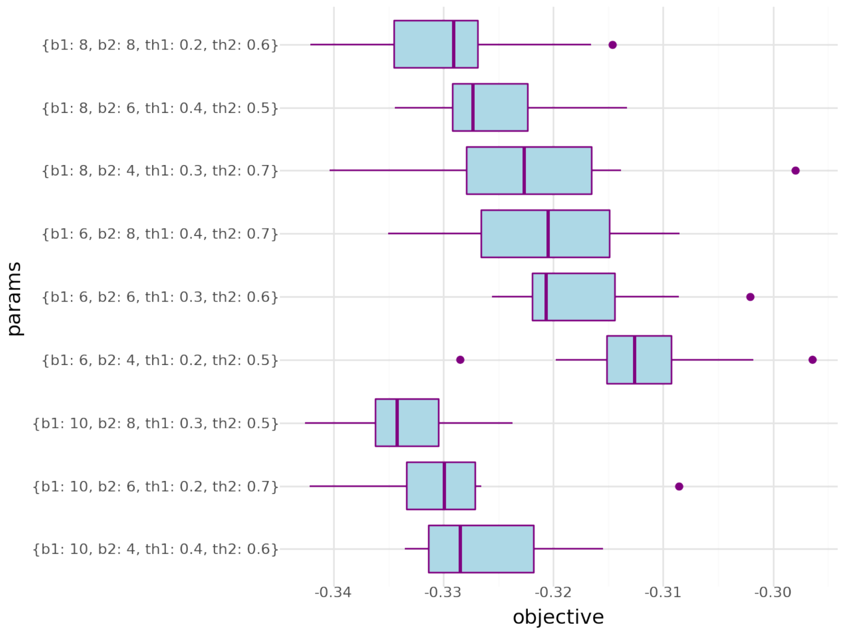

Figure 6.10: Boxplots of objective values of hyperparameter sets SS

7

Evaluation of Results

This chapter discusses the experimental phase of the research. The experiments are executed using a Python 3.12.3 engine in Visual Studio Code, on a computer with an Intel(R) Core(TM) i5-1145G7 CPU of 2.60Ghz and 16.0GB of RAM. Section 7.1 discusses the verification of the models. The models will be validated in Section 7.2. Section 7.3 compares and discusses the best solutions found by MILP, BRKGA, and SS. Subsequently, a series of experiments will be conducted to address the main research question and its sub-components. The experimental structure is as follows:

- Section 7.4 will present and analyze the results from experiments investigating the impact of increasing instance size.
- Section 7.5 will assess the effects of varying the starting time windows.
- Section 7.6 will assess the effects of business rules on both the convergence speed and the solution quality.

In each experiment, four distinct model configurations will be employed to solve the problem instances:

- BRKGA without local search
- BRKGA-LS with local search
- SS without local search
- SS-LS with local search

The results of these experiments will be analyzed to provide a well-founded answer to the main research question:

Does the implementation of local search boost the performance of meta-heuristics in solving the Consultant-to-Project Assignment (C2Pa) problem at BearingPoint NL, specifically focusing on efficiency and efficacy?

7.1 Verification of Models

Before the experiments can be executed, it should first be proven that the C2Pa models work as intended. In the process of proving the correctness, we will generate an easy test instance, solve it with the BRKGA model, and compare the returned KPI values with manually calculated ones. These values should be identical if the model behaves as expected. The problem will be simplified to only take one KPI into account, which is done by setting the targeted KPI with a value of 1 and the others with 0. This simplifies the optimization problem and allows us to closely monitor the behavior of the model. For each KPI an experiment will be conducted. The instance contains one project from the D&A service line. It involves nine skills, has a duration of five weeks and requires one consultant. Moreover, the project should start in week 36 and has no flexible time window.

7.1.1 Skill Match KPI

Table 7.1 shows the results of solving the instance when only using the skill match KPI. This result implies that on average, one skill is negatively mismatched, thus indicating underqualification. Given the availability of the practice, only consultants 66 and 69 are available to be assigned to this project. To determine whether the actions of the model are in line with the expectations, the skill match between the project and these consultants must be calculated. The required skill level of the nine skills of the project and the consultant skill levels on these skills are shown in Table 7.3. By using Equation 6.3 for the calculation of the skill match, we obtain the unnormalized skill match, as can be seen in Table 7.2. The algorithm uses the normalized skill match in the evaluation of the fitness. The normalizing constant is calculated as the maximum skill level times the number of skills, and this value divided by the number of projects. Executing this calculation results in a normalizing constant of 81. Dividing the skill match by this constant results in the skill match that is used in the fitness calculation. For consultant 66 this is -0.111 and for consultant 69 this is -0.234. From Equation 6.8 can be concluded that the algorithm will choose the skill match that is closest to 0, meaning that consultant 66 will be assigned to the project. Dividing the unnormalized skill match of consultant 66 by 9 (since the skill match is negative) gives us the average project skill mismatch of -1. This value is identical to the skill match presented in Table 7.1, which shows that the skill match KPI works correctly.

KPI	value
Skill Match	-1
Utilization	1.0
Satisfaction	6.444
Hourly cost	127.0

Table 7.1: Skill match verification results

	C66	C69
Unnormalized Skill match	-9	-19
Normalized Skill match	-0.111	-0.234
Average mismatched skills	-1	-2.111
Assigned consultant	✓	×

Table 7.2: Skill match C66 and C69

	Skill 1	Skill 8	Skill 19	Skill 104	Skill 105	Skill 108	Skill 115	Skill 123	Skill 125
PSL	3	1	1	3	1	1	3	1	3
C66	3	2	2	2	3	2	2	2	2
C69	3	2	1	2	1	0	2	1	0

Table 7.3: Required skill level and consultant skill level

7.1.2 Satisfaction KPI

The same instance will be employed to confirm the models' behavior concerning the satisfaction KPI. As the instance remains unchanged, consultants 66 and 69 continue to be the only available options. Table 7.4 displays the outcomes of solving the instance while solely focusing on the satisfaction KPI. The goal is to verify if manual computations will also yield a satisfaction score of 7.889. Calculating the satisfaction KPI for a single project simply involves determining the mean satisfaction score of the relevant skills. Table 7.5 illustrates the satisfaction scores of the involved skills for both consultants. The average satisfaction scores for consultants 66 and 69 are 6.444 and 7.889, respectively. Since the objective is to maximize this value, consultant 69 will be assigned to the project. This justifies the model's behavior regarding the satisfaction KPI.

KPI	value
Skill Match	-2.111
Utilization	1.0
Satisfaction	7.889
Hourly cost	120.0

Table 7.4: Satisfaction verification results

	Skill 1	Skill 8	Skill 19	Skill 104	Skill 105	Skill 108	Skill 115	Skill 123	Skill 125
C66	10	6	4	7	9	1	6	8	7
C69	10	7	5	8	9	7	9	9	9

Table 7.5: Satisfaction scores of consultants 66 and 69

7.1.3 Hourly Cost KPI

Verifying the behavior of the hourly cost KPI is straightforward, check whether the consultant with the lowest hourly cost is assigned to the project. The KPIs returned by the model can be found in Table 7.6. The hourly cost of 120 indicates that the assigned consultant has the job position of a senior management analyst. By solving the same instance as before, the pool of available consultants is still restricted to only consultants 66 and 69. The job positions of these consultants are consultant and senior management analyst, respectively. A consultant has a hourly cost of 127, whereas a senior management analyst has a hourly cost of 120. Given the positive sign of this KPI in the fitness function stated in Equation 6.8, the goal is to minimize this value. This results in assigning consultant 69 with the job position of senior management analyst and an hourly cost of 120, which is in line with the model result. Therefore, the hourly cost KPI is working as expected.

KPI	value
Skill Match	-2.111
Utilization	1.0
Satisfaction	7.889
Hourly cost	120.0

Table 7.6: Hourly cost verification results

7.1.4 Utilization KPI

The utilization KPI has been introduced to give prioritization to client projects over internal projects. Furthermore, BE NL has also indicated that full-time projects are favored over part-time projects and that long-term projects are preferred over short-term projects. Each scenario will be examined with a unique test instance to verify the behavior of the model. In each scenario, although the utilization KPI value is positive, it is subtracted in the overall objective function. Therefore, the optimization goal for this specific KPI is maximization, which contributes to minimization of the fitness function. The objective is calculated as the total hours of the accepted client projects divided by the total hours of all the projects (client, beach, and declined projects).

Client projects over internal projects Accepting client projects over internal projects is an essential rule that is the underlying assumption of the two other rules. To verify that this rule is enforced, a test instance is generated where two identical projects need to be scheduled. One project is a client project, whereas the other is an internal project. The availability of consultants has been modified so that only consultant 66 is available. The required utilization of the projects only allows the acceptance of one of the two projects. Table 7.7 shows the results of solving this instance. If we were to accept the beach project and decline the client project, the contribution of client projects hours would result in 0. If we were to accept the client project, $pt_1 \times ph_{1,0}$ client hours are accepted. Recall that the duration and utilization of the projects are identical. This means that the accepted client hours are responsible for half of the total hours, which is 0.5. Since this KPI needs to be maximized, the model should choose the second scenario with a utilization of 0.5. This is indeed the case, as can be seen in Table 7.7, which verifies the behavior of the model.

KPI	value
Skill Match	-1
Utilization	0.5
Satisfaction	6.444
Hourly cost	127.0

Table 7.7: Verification client projects over internal projects

Full-time over part-time To verify that full-time projects are chosen over part-time projects, a test instance of two projects is generated where consultant 66 only has the availability to be assigned to one of the two. The requirements of the projects are identical, however, the utilization of the consultant on project 1 is 100% whereas project 2 only requires 50% utilization of a consultant. Utilization of 100% corresponds to 40 working hours. The ratio of total projects hours between the two projects is 2:1. If project 1 is accepted, the utilization KPI has a value of $40/(40+20) = 0.667$. If project 2 is accepted, this value is $20/(40 + 20) = 0.334$. As the objective of this KPI is maximization, the first project should be accepted, resulting in an utilization KPI of 0.667. This is in line with the best result of the model presented in Table 7.8. With that, the model's behavior regarding full-time vs part-time has been verified.

KPI	value
Skill Match	-1
Utilization	0.667
Satisfaction	6.444
Hourly cost	127.0

Table 7.8: Verification full-time projects over part-time projects

Long-term over short-term Lastly, it has to be verified to the model chooses long-term projects over short-term projects. Again, a test instance of two identical projects is generated where only consultant 66 has the availability to be assigned to one of the two. The lengths of the projects are as follows, project 1 has a length of 15 weeks, whereas project 2 has a length of 5 weeks. The utilization of roles of the projects are the same, so these do not have to be taken into account in the objective. If project 1 accepted, the KPI has a value of $15/(15 + 5) = 0.75$. If project 2 is accepted, this value is $5/(15 + 5) = 0.25$. The model should accept project 1, as this KPI value is the highest. Table 7.9 shows that the model solved the test instance with a resulting utilization KPI of 0.75. Since these solutions are identical, the model's behavior concerning long-term vs short-term has been verified.

KPI	value
Skill Match	-1
Utilization	0.75
Satisfaction	6.444
Hourly cost	127.0

Table 7.9: Verification long-term projects over short-term projects

7.2 Validation of Models

Besides verifying that the model makes the right decisions, it is also essential to assert that the solutions of the model are compliant with the rules and constraints of the problem. The model is validated by utilizing the existing MILP model. Since the MILP model of (1) is validated for correctness and compared to the OTL, if an approach is found that validates our model with the MILP, it is proven that the model is compliant with the rules and constraints of the C2Pa problem.

The MILP model has been restructured to function as a feasibility check for solutions generated by the metaheuristic models. The feasibility of a solution depends solely on the decoder, as solutions are constructed within this component. Consequently, the specific model used to submit the solution to the restructured MILP is irrelevant for the feasibility assessment. The solution requiring validation originates from the BRKGA model. Importantly, if this BRKGA-derived solution is considered feasible, it logically follows that solutions produced by Scatter Search must also be feasible.

In this validation approach, all decision variables in the MILP model are fixed to the values obtained from the BRKGA solution. In doing so, the MILP model is transformed from an optimization problem into a constraint satisfaction problem. This modified MILP model acts as a feasibility checker. When we input the BRKGA solution into this fixed-variable MILP model, it allows us to verify whether the solution satisfies all constraints and rules of the C2Pa problem. If the MILP solver can find a feasible solution with these fixed variables, it confirms that the BRKGA solution is indeed valid and compliant with all problem constraints.

The test instance generated for this experiment contains a single project that requires three consultants, has a duration of 12 weeks, and involves 21 skills. This instance has sufficient complexity, as it includes assigning multiple consultants and division of skills over these consultants. The numerical results are not of importance in this situation; the point of interest is whether the MILP model will classify the solution as feasible or infeasible. After solving the instance using the BRKGA model and submitting the solution to the restructured MILP model, the Gurobi solver confirmed the feasibility of the solution. Moreover, the resulting recalculated values of the KPIs are identical. This result validates both the models and, more specifically, the decoder algorithm used within them.

7.3 Comparison to MILP

This section compares the optimal solution obtained by the MILP model with the best solution found by the BRKGA and Scatter Search models. It also examines the total computation time required for each model, from initialization to completion. The comparison of computation times demonstrates the relative efficiency of the BRKGA and Scatter Search models compared to the MILP model. In particular, this analysis includes the time required for model construction, which is found to be significant for MILP. The performance comparison reveals whether both models arrive at similar solutions or if there are discrepancies in their problem-solving dynamics, potentially due to differences in their respective objective functions.

The test instance that will be used is made up of two projects. The first project must start at week 20 and has no flexible starting window. It has a duration of 13 weeks, requires one consultant on a 16-hour basis per week, and involves four skills. The second project should start between week 34 and week 36. It has a duration of 10 weeks, requires two consultants on a 20-hour and 36-hour basis per week, and involves 10 skills where the first consultant needs to fill 4 skills and the second consultant 6 skills. All three models are configured to use the weights determined in Section 6.3.5, thus following the preferences of the OTL. BRKGA and Scatter Search are executed for 50 generations each with their best hyperparameters as found in Section 6.5.2 and 6.6.2. Local search was omitted from this experiment due to two primary factors. Firstly, the problem instance is relatively small in scale. Secondly, preliminary analyses indicated rapid convergence of the solution.

The results of the three models can be found in Table 7.10, 7.11 and 7.12. It can be seen that BRKGA and Scatter Search produced the same optimal solution. Despite showing similarities, there are significant differences. First of all, the total time that the MILP model needs to initialize and solve the problem is almost 8 times greater than that of the BRKGA model. The computation time of Scatter Search should be slightly reduced to make the comparison fair, as one generation of SS produces more solutions per generation compared to BRKGA. Secondly, the values of the KPIs are not identical between MILP and the meta-heuristics. This difference is probably due to the objective functions. The meta-heuristics calculate the fitness value directed from the KPIs, whereas the MILP model uses the deviation variables in combination with aspiration-level variables. The aspiration-level variables act as penalties when the desired variables deviate from their desired level.

These penalties could influence the behavior of the model. Moreover, no request from the OTL was done to include these penalties in the models of the meta-heuristics.

KPI	value
Skill Match	-0.944
Utilization	1.0
Satisfaction	4.539
Hourly cost	139.737
Time	351.66 s

Table 7.10: Results of MILP

KPI	value
Skill Match	-0.222
Utilization	1.0
Satisfaction	4.447
Hourly cost	143.026
Time	44.95 s

Table 7.11: Results of BRKGA

KPI	value
Skill Match	-0.222
Utilization	1.0
Satisfaction	4.447
Hourly cost	143.026
Time	48.49 s

Table 7.12: Results of Scatter Search

Given the preferences and rules provided by the OTL, the solutions of BRKGA and Scatter Search are more in line with the instructions by favouring the skill match KPI. The skill match KPI of the metaheuristics is closer to 0 with the -0.222 average mismatched skills per project, compared to the -0.944 of MILP. This is achieved by slightly compromising the satisfaction and the hourly cost KPI compared to MILP. To be able to compare the quality of the solutions based on one value, the MILP solution has been transformed to the decision variable of the BRKGA and Scatter Search model. Calculating the fitness value of MILP using this approach resulted in -0.2760, whereas the solution of BRKGA yielded -0.3186. This comparison is not completely fair, as MILP introduces deviation penalties on the hourly cost, utilization, and satisfaction variables. This concludes that the inclusion of aspiration intervals in the objective function significantly influences the problem-solving dynamics of the optimization model.

7.4 Experiment Instance Size

This section will evaluate the effectiveness of the proposed models as problem instance size increases. The analysis aims to identify the limits of these models. Additionally, it will assess the impact of incorporating local search techniques into the models. To address this research question, problem instances will be solved using both versions of the models: one with local search enabled and another with it disabled. For each model configuration, 15 independent runs will be conducted, with results recorded for both performance and computational speed. These experimental outcomes will contribute significantly to answering the main research question, providing insights into the scalability and efficiency of the proposed approaches.

The problem instances under examination include portfolio sizes of 3, 6, 12, and 24 projects. Each instance has been designed to include sufficient complexity, i.e., each instance will contain projects that require multiple team members and a significant amount of skills. Note that not every project has this level of complexity. To ensure fair comparisons between the models, several measures have been implemented. Firstly, the number of generations in the base models has been calibrated to produce a similar number of offspring. Analysis revealed that on the base instance of 3 projects, the SS algorithm generates 204 children per generation, while BRKGA produces an average of 60. To maintain fairness, the ratio of generations between BRKGA and SS should be set at approximately 3.4:1. Secondly, a standardised approach has been implemented to equalize the total number of solutions subjected to local search between the two models.

The frequency of local search will be adjusted as the instance size increases. Local search frequency adaptation is crucial when scaling genetic algorithms to larger problem instances, as it helps maintain an effective balance between diversification and intensification. Following Eiben and Smith (36), we adapt the local search frequency using a non-linear scaling equation that prevents excessive computational overhead as the instance size grows:

$$f_{ls} = \lfloor f_{base} \times \sqrt{\frac{p}{p_{base}}} \rfloor \tag{7.1}$$

As stated earlier, the base frequency f_{base} is set to 10 and the base instance size p_{base} to 3. These adjustments aim to create equal conditions for comparing the performance of the SS and BRKGA models across different problem sizes, ensuring that any observed differences can be attributed to the algorithms' fundamental capabilities rather than differences in their parameters.

The research process begins with a benchmark experiment to evaluate how quickly the local search variant reaches convergence. By analyzing the convergence curves, we will identify the specific number of generations needed for this process. This same generation count will then serve as a parameter for running the base models, allowing for direct comparison. The next phase involves statistical analysis of the objective values obtained from both models. When statistical tests reveal significant differences between a local search variant and its base counterpart, we move to an additional experiment. This additional experiment is necessary because local search variants typically require more evaluations of the fitness function.

To ensure a fair comparison, we will adjust the number of generations to achieve matching CPU times between models. The adjustment will be calculated using the time percentage increase between variants. This methodology, which follows Lobo et al.'s (37) research framework, provides a fair basis for comparing algorithms with different computational demands per generation. The final statistical analysis of these results will definitively demonstrate whether local search offers meaningful benefits in each specific instance.

7.4.1 Instance size 3

The first experiment consists of an instance of three projects, chosen to represent three service lines in the instance. The projects require 1, 1, and 3 consultants, and the involved skills total 4, 13, and 24. Together, they are responsible for a chromosome length of 206, indicating an overload of consultants available for the projects. Prior analysis on the convergence curves of both models indicated that the BRKGA models should run for 275 generations and the SS models for 80 generations. The curves are presented in Figures 7.1 and 7.2. The random number seed is fixed for every problem to ensure identical starting points. In Figure 7.1, the progression of the best fitness and the average fitness of the elite of the BRKGA models over generations is shown. The peaks in the average fitness curves of both models represent the shaking events. These could be caused by an identical elite or by not improving the best solution for R generations. The figure suggests that local search is not inherent in faster convergence, as BRKGA and BRKGA+LS go head-to-head for almost the entirety of the run. In Figure 7.2, the progression of the best fitness and the average fitness of the reference set B_1 of the SS models over generations is shown. The behavior of the best and average fitness appears almost identical, although SS+LS having a minor advantage throughout the run. In both figures, the best fitness lines are closely followed by the average fitness lines, indicating that improvements are successfully being transferred within the population. The convergence curves of BRKGA and SS show that the models converged well before the maximum number of generations. However, the best fitness of the BRKGA models do not align, whereas they do align at the SS models.

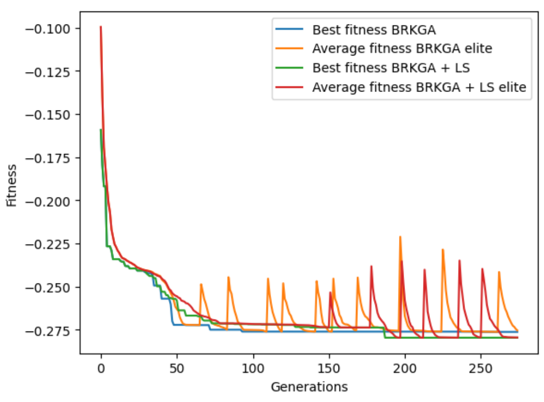

Figure 7.1: Convergence curve BRKGA vs BRKGA+LS of instance size 3

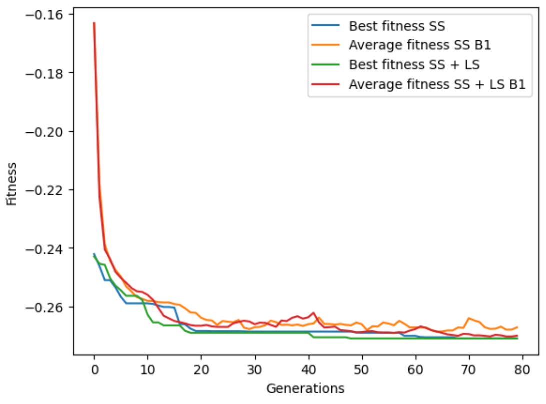

Figure 7.2: Convergence curve SS vs SS+LS of instance size 3

Table 7.13 presents the performance metrics of the benchmark experiment of 15 independent runs. The models that include local search are indicated by the added '+LS'. The table shows that there is no clear difference among the results of the models. The addition of local search to the models did slightly improve the overall solution quality of the models of this instance. For both models, local search decreased the average objective value and managed to get the average mismatched skills closer to 0. Three out of the four models achieved the same highest score. The BRKGA model seems to be lacking in terms of performance compared to the other models. More on the effects of introducing local search, the computational time needed to solve the instance increased by 68.00% for BRKGA and 100.22% for SS. Based on the results in this table, local search seems to deliver minor improvements while significantly sacrificing the speed of the algorithm compared to the base models.

	BRKGA	SS	BRKGA+LS	SS+LS
best objective	-0.2776	**-0.2796**	**-0.2796**	**-0.2796**
average objective	-0.2724	-0.2741	-0.2740	**-0.2759**
time (sec)	38.50	27.48	64.68	55.02
time increase (%)	-	-	68.00%	100.22%
average mismatched	-2.9037	-2.8716	-2.8098	**-2.7827**
average satisfaction	6.592	**6.614**	6.529	6.581
average cost	137.06	**136.35**	137.98	137.54

Table 7.13: Results experiment instance size 3

Figure 7.3 presents the distribution of the objective values for each model in the form of boxplots. For both BRKGA and SS algorithms, the local search variants tend to achieve slightly lower median objective values, suggesting that local search generally improves solution quality. Furthermore, the first three models show variability in their distribution. SS shows comparable variability to BRKGA, whereas BRKGA+LS demonstrates slightly more variability. SS+LS shows the smallest interquartile range, indicating more consistent performance, although it does have outliers.

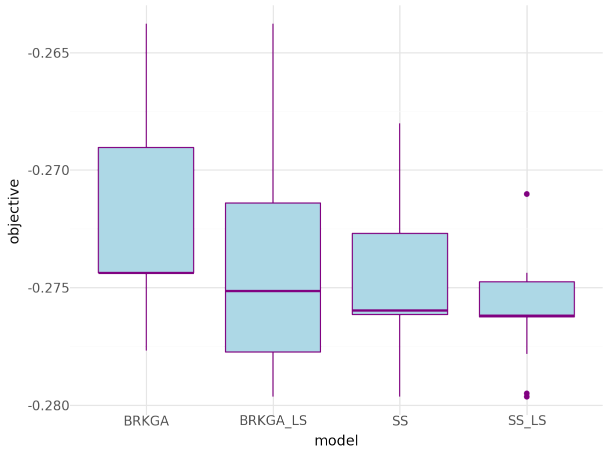

Figure 7.3: Boxplots of objective values of the models of instance size 3

In this and future experiments, statistical tests will be performed to provide evidence that one model outperforms the other. These tests help ensure that observed differences between models are not due to random variation. First, the distribution of the data must be checked to determine whether it follows a normal distribution, which influences the choice between using a t-test or a Wilcoxon Signed-Rank test. The Shapiro-Wilk test used to determine whether the data is normally distributed. If the data is normally distributed (p-value $\geq$ 0.05) , a t-test will be used, while the Wilcoxon Signed-Rank test will be applied if the data is not normally distributed (p-value $<$ 0.05). Both tests assess the null hypothesis (H_0), which assumes that there is no significant difference in the mean performance between the two models. The null hypothesis suggests that adding Local Search has no effect on the models' mean performance. The tests will evaluate whether the null hypothesis can be rejected based on the p-value. If the p-value is less than 0.05, the null hypothesis is rejected, indicating that the model with Local Search significantly outperforms the other in terms of mean performance. If the p-value is greater than or equal to 0.05, the null hypothesis cannot be rejected, implying insufficient evidence to claim a difference in the mean performance of the models.

The Shapiro-Wilk test concluded that the objective values of SS are not normally distributed. Consequently, the BRKGA models will perform the t-test while the SS models will be exposed to the Wilcoxon Signed-Rank test to determine whether local search variants outperform their base variants. The statistical test between BRKGA and BRKGA+LS resulted in a p-value of 0.3524, where the test between SS and SS+LS resulted in a p-value of 0.1728. Since both p-values are greater than 0.05, the null hypothesis cannot be rejected, implying insufficient evidence to claim a difference in the mean performance of the models. This result is in line with the discoveries found in Table 7.13 and Figure 7.3. Because no statistically significant differences have been found, no additional experiments will be performed. Although not significantly different, the p-value of 0.1728 of SS+LS is not far away from the threshold of 0.05. In addition to that, Figure 7.3 indicates that the SS+LS model produces the most reliable results while maintaining high-quality solutions.

7.4.2 Instance size 6

The second experiment consists of an instance of six projects, chosen to represent all four service lines in the instance. The projects require 1, 2, 1, 1, 4 and 4 consultants, and the involved skills total 3, 10, 9, 17, 47 and 35. Together, they are responsible for a chromosome length of 447, which is more than double compared to the previous experiment. The local search frequency of this experiment is determined using Equation 7.1, which produced a frequency f_{ls} of 14.

Figures 7.4 and 7.5 present the convergence curves of the models when starting with identical solutions. Figure 7.4 suggests that the curves have plateaued. Figure 7.5 shows that there could be room for minor improvements for SS as we approach the maximum number of generations. The curves have sufficiently plateaued to justify the choice of 550 generations for the BRKGA models and 250 generations for the SS models. Another conclusion from the curves is that not all local search improvements contribute to a permanent advantage over the base variant. In Figure 7.4 it can be seen that in generation 55 a great improvement was made; however, BRKGA managed to almost close the gap to BRKGA+LS after 200 generations in terms of the best fitness value. It can be seen that the curves of both models have converged around the same level. The curve of BRKGA+LS converged much sooner than the maximum number of generations, which triggered a shaking event. Figure 7.5 clearly presents the events of local search improvements. These improvements in generation 50 and 75 resulted in an advantage for SS+LS over SS, however, this advantage is slowly getting smaller as SS+LS has converged around generation 125 and was stuck in a local optima.

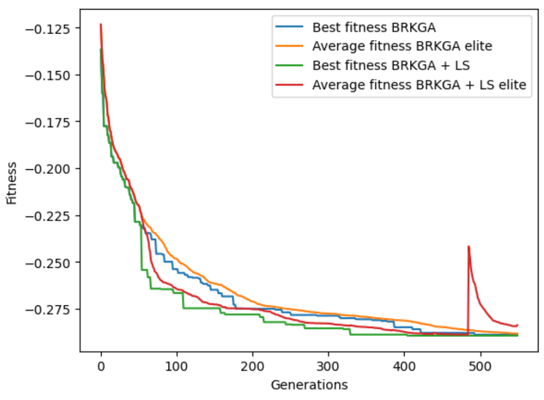

Figure 7.4: Convergence curve BRKGA vs BRKGA+LS of instance size 6

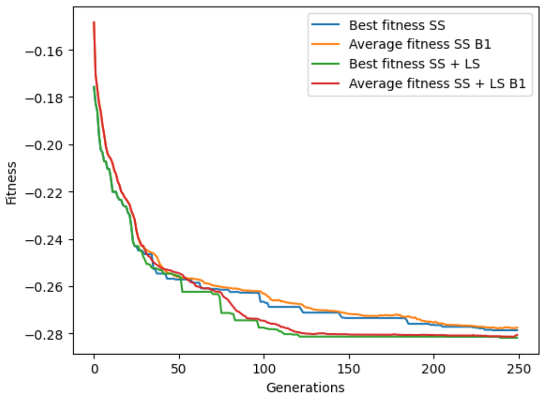

Figure 7.5: Convergence curve SS vs SS+LS of instance size 6

The results of this experiment are presented in Table 7.14. This time, clear differences can be found between the models. First, the base BRKGA model is not on par with the other models in terms of solution quality. It achieved the worst average objective value and the worst best objective value. The SS+LS algorithm achieved the highest best objective value and highest average objective value. For both models, the local search variants obtained a considerably higher average objective value, however, this is accompanied by an average time increase of 169.29% and 181.69% for BRKGA+LS and SS+LS.

	BRKGA	SS	BRKGA+LS	SS+LS
best objective	-0.2871	-0.2917	-0.2915	**-0.2980**
average objective	-0.2760	-0.2795	-0.2828	**-0.2862**
time (sec)	168.48	193.55	453.72	545.41
time increase (%)	-	-	169.29%	181.69%
average mismatched	-3.3456	-3.1950	-3.1679	**-3.1012**
average satisfaction	5.774	5.756	5.909	**6.017**
average cost	142.60	141.91	141.63	**141.40**

Table 7.14: Results experiment instance size 6

Figure 7.6 shows the distribution of the objective values of this experiment. It confirms the claim that the base BRKGA model is lacking. The spread of BRKGA+LS is most consistent over the runs. The BRKGA models are the only models that produced no outliers. The performance of SS+LS is suppressed by huge outliers. Without the outliers, the performance of SS+LS would be even more superior compared to the other models.

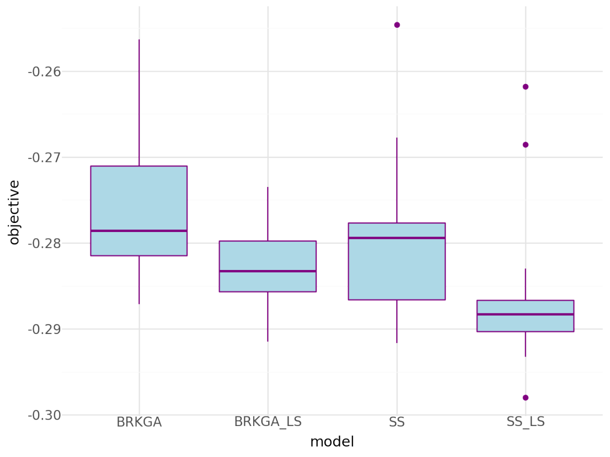

Figure 7.6: Boxplots of objective values of the models of instance size 6

The normality test concluded that all models but SS+LS produced normally distributed objective values since the p-values of the first three models are greater than 0.05, whereas the p-value of SS+LS is $0.0017 < 0.05$. This allows the t-test to be used for the BRKGA models and Wilcoxon Signed-Rank test for the SS models to test whether or not local search variants have a different mean objective score than base models. The t-test of BRKGA vs BRKGA+LS resulted in a p-value of 0.0137, which is smaller than 0.05. This provides enough evidence to reject the null hypothesis that the means are the same, thus indicating that the objective values from BRKGA and BRKGA+LS are significantly different. The Wilcoxon Signed-Rank test of SS vs SS+LS resulted in a p-value of 0.0301. The null hypothesis is rejected, as $0.0301 < 0.05$. The differences in objective values are statistically significant, hence SS+LS outperforms SS.

Additional experiments are required to assess the difference in performance between the BRKGA models and the SS models, this time BRKGA and SS receive as much CPU time as BRKGA+LS and SS+LS. Using the percentage time increase as multiplier, BRKGA should run for $550+550\times1.6929 \approx 1480$ generations and SS should run for $250+250\times1.8169 \approx 705$ generations. Table 7.15 presents the results of the fair experiments.

Unlike earlier, the quality of the solutions of BRKGA and has now surpassed BRKGA+LS. The Shapiro-Wilk test concluded that the distribution of BRKGA is normally distributed. The subsequent t-test between BRKGA and BRKGA+LS produced a p-value of 0.7730, indicating that the distributions are not statically significantly different. Thus, for a problem of six projects with equal conditions, incorporating local search in BRKGA does not result in statistically significant improvements. Moreover, under the same conditions, BRKGA has a slight advantage over BRKGA+LS. Table 7.15 reveals that SS is by far the best performing model. Statistical tests are now used to assess whether SS is superior to SS+LS. Since SS+LS was not normally distributed, the Wilcoxon Signed Rank test is used. This test yielded a p-value of 0.4210, indicating that there is insufficient evidence to claim a difference in the mean performance of the SS models. However, SS has shown to have a substantial advantage over SS+LS. So, it shows that using far more generations than intended should give more benefits than using local search.

	BRKGA	BRKGA+LS	SS	SS+LS
best objective	-0.2952	-0.2915	-0.2979	-0.2980
average objective	-0.2835	-0.2828	-0.2905	-0.2862
time (sec)	404.92	453.72	457.34	545.41
average mismatched	-3.0728	-3.1679	-2.8901	-3.1012
average satisfaction	5.868	5.909	6.028	6.017
average cost	143.32	141.63	142.73	141.40

Table 7.15: Results additional experiments instance size 6

7.4.3 Instance size 12

The third experiment consists of an instance of 12 projects. The portfolio of projects consists of six projects that require one consultant, three projects that require two consultants, two projects that require three consultants, and a single project that requires four consultants. In terms of involved skills, the portfolio consists of projects of mixed complexity as the skills total 3, 10, 9, 17, 47 and 35. Together, they are responsible for a chromosome length of 869, which doubles the length of the chromosome compared to the previous experiment. The local search frequency of this experiment is determined using Equation 7.1, which produced a frequency f_{ls} of 20.

Figures 7.7 and 7.8 show the final convergence curve after investigating the behavior of this instance. From the figures can be concluded that the curve of BRKGA+LS has flattened around 1100 generations, whereas the curve of SS+LS converged around 500 generations. Besides that conclusion, it can be seen that for this run the best and average fitness of the SS+LS model is marginally better than the SS model throughout its run. This cannot be said about BRKGA+LS, as the inverse is true. While starting with the same solutions, BRKGA managed to dominate over BRKGA+LS. Whether these results are caused by variance will be determined in a second.

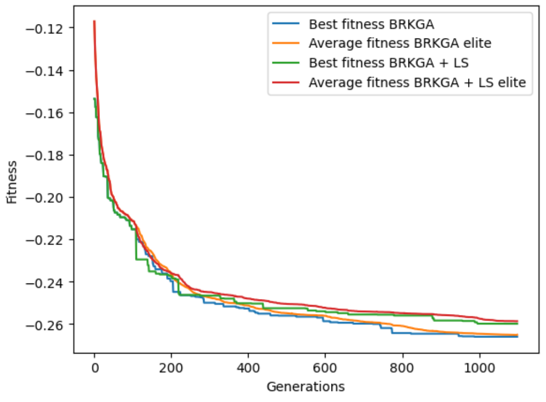

Figure 7.7: Convergence curve BRKGA vs BRKGA+LS of instance size 12

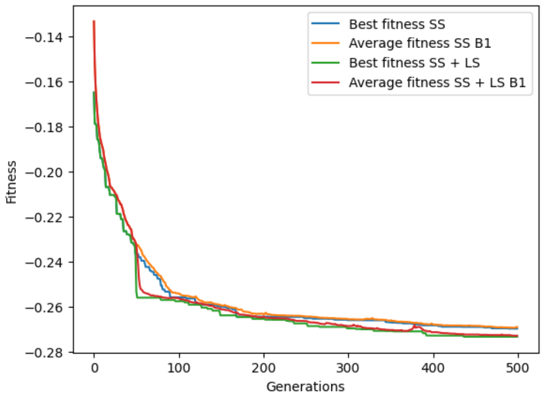

Figure 7.8: Convergence curve SS vs SS+LS of instance size 12

Table 7.16 displays the results of the experiment. As in the previous experiment, there is a gap in solution quality between BRKGA and BRKGA+LS. Moreover, the BRKGA model lacks performance in all metrics except for hourly cost compared to BRKGA+LS. The aggregated results show that SS has a slight advantage over SS+LS in terms of average objective, average mismatched skills, and average cost. Remarkably, SS managed to dominate both local search models despite needing to solve a quite complex instance. Despite that, the best objective has been achieved by BRKGA+LS. Compared to SS+LS, BRKGA+LS starts to show signs that it is less effective in terms of average objective value. Introducing local search increased the average time 109.01% and 107.76%, which is relatively significantly lower than in the previous experiment despite having a larger instance. Moreover, local search is not inherent to superior solutions. A separate experiment with the BRKGA model has been performed to demonstrate the difference in computational requirements of the decoder and the evaluation of KPIs. After disabling the calculation of the KPIs, the model finished the experiment in 293.80 seconds. This revealed that the main method plus the decoding process contributes to 54.51% of the total computational time, while the evaluation of the KPIs uses the remaining 45.49% of the computational time.

	BRKGA	SS	BRKGA+LS	SS+LS
best objective	-0.2686	-0.2734	**-0.2767**	-0.2737
average objective	-0.2560	**-0.2663**	-0.2636	-0.2661
time (sec)	539.00	556.62	1126.57	1156.42
time increase (%)	-	-	109.01%	107.76%
average mismatched	-2.8444	-2.5284	-2.5784	**-2.4722**
average satisfaction	5.578	**5.734**	5.667	5.665
average cost	**138.49**	140.91	141.36	142.26

Table 7.16: Results experiment instance size 12

The boxplots in Figure 7.9 further elaborate on the spread of the objective values of the models. It is clearly visible that BRKGA is not on par with the other models. The figure demonstrates the claims made above, despite SS not achieving the best score, the values are more clustered compared to the local search models, and by that, a better average score is achieved. So although it is not the best model in terms of best objective, it produces more reliable results than the other models. The boxplots also show that the produced objective values of BRKGA+LS are wildly inconsistent, producing values ranging from the worst objective to the best objective.

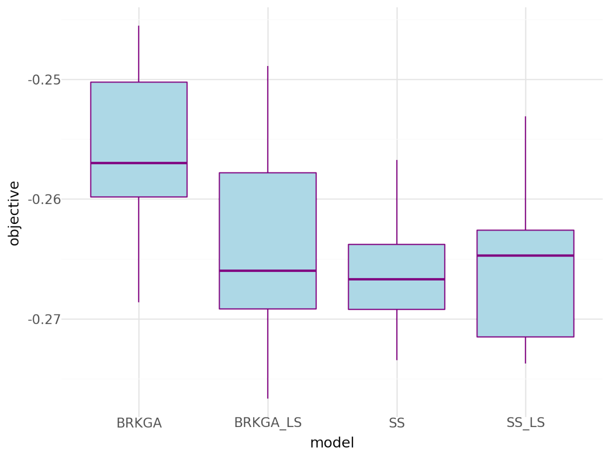

Figure 7.9: Boxplots of objective values of the models of instance size 12

Statistical tests should reveal whether the differences in the models are statistically significant. First, the Shapiro-Wilk test indicated that the objective values of all models are normally distributed. Therefore, both models will perform the t-test. The subsequent t-test of the BRKGA models resulted in a p-value of 0.0126, this value rejects the null hypothesis and thus indicates that the mean objective value of the BRKGA+LS model is different from the mean objective value of BRKGA, i.e. the BRKGA+LS model outperforms the BRKGA model when using the same amount of generations. The t-test of the Scatter Search models yielded a p-value of 0.9183. The p-value is greater than 0.05, so the null hypothesis cannot be rejected, implying that there is not enough evidence to claim that the mean objective values of SS and SS+LS are different. These conclusions are in line with the claims made based on Table 7.16 and Figure 7.9. However, it is still quite noteworthy that SS seems to be the overall best-performing model despite not utilizing any solution improvement method and having a significantly smaller number of fitness evaluations compared to SS+LS.

An additional experiment is required to reveal whether the BRKGA model is also capable of approaching the solution quality of its local search variant. In this experiment, BRKGA receives equal CPU time as the BRKGA+LS model. Using the percentage time increase as multiplier, the BRKGA should run for $1100 + 1100 \times 1.0901 \approx 2300$ generations. Table 7.17 shows the results of the fair comparison between BRKGA and BRKGA+LS. As in the previous experiment, the base variant surpasses the local search variant in average objective value when given the same computational time. Given the results, a statistical test would not add much value. This is confirmed as the Wilcoxon Signed-Rank test produced a p-value of 0.5995, demonstrating that there is no significant difference in means between the two models.

	BRKGA	BRKGA+LS
best objective	-0.2746	-0.2767
average objective	-0.2644	0.2636
time (sec)	1091.38	1126.57
average mismatched	-2.6099	-2.5784
average satisfaction	5.745	5.667
average cost	140.58	141.36

Table 7.17: Results additional experiment instance size 12

7.4.4 Instance size 24

The last experiment consists of the complete project portfolio, i.e., 24 projects. Chapter 4 provides an extensive description of the projects. After encoding the portfolio, the length of the chromosome is 1551. In previous experiments, the convergence speed slowed down by more than half when the chromosome length doubled. Since the chromosome length has now doubled compared to the previous experiment, the number of generations has been set to 2500 for BRKGA and 1100 for SS. The convergence curves in Figures 7.10 and 7.11 show that in terms of computational budget, BRKGA+LS had sufficient generations as its best fitness was found 1000 generations before reaching the maximum number of generations. A shaking event has event occurred because of the inability to improve its best fitness score. Analysis of Figure 7.11 reveals an alternating pattern between SS and SS+LS throughout their execution. While both models achieve comparable final fitness values, the convergence curves indicate that neither has reached a definitive plateau. This suggests that additional improvements might be possible with an extended computational budget. However, to maintain consistency and fairness in comparison with the other models, the decision was made to maintain the current computational limitations. The local search frequency of this experiment is determined using Equation 7.1, which produced a frequency f_{ls} of 28.

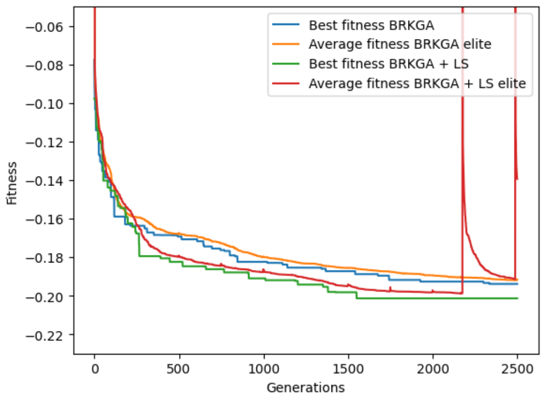

Figure 7.10: Convergence curve BRKGA vs BRKGA+LS of instance size 24

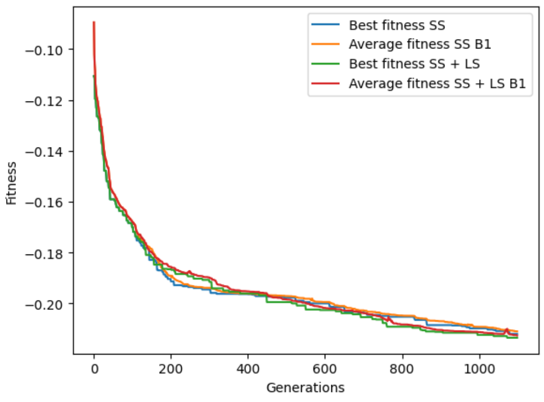

Figure 7.11: Convergence curve SS vs SS+LS of instance size 24

The experimental results, presented in Table 7.18, demonstrate patterns consistent with previous findings. The Scatter Search models significantly outperform both BRKGA variants, with BRKGA showing notably inferior results. For the BRKGA models, the introduction of local search resulted in better results in every metric. In addition to this, the best objective found by BRKGA+LS is at a distance from the best objective found by BRKGA. For Scatter Search, the implementation of local search shows moderate impact, with aggregated results revealing minor improvements in every metric. The computational overhead introduced by local search is now showing increases of 108.38% for BRKGA and 102.85% for SS. Therefore, the improvements come with a huge sacrifice in terms of required computational time.

	BRKGA	SS	BRKGA+LS	SS+LS
best objective	-0.2022	-0.2172	-0.2220	**-0.2227**
average objective	-0.1962	-0.2064	-0.2007	**-0.2090**
time (sec)	2251.87	2329.96	4692.53	4726.32
time % increase	-	-	108.38%	102.85%
average mismatched	-4.3617	-4.135	-4.2775	**-4.0549**
average satisfaction	4.721	4.907	4.813	**4.922**
average cost	137.73	136.79	**136.65**	136.71

Table 7.18: Results experiment instance size 24

The distributions of the objective values of the models are presented in Figure 7.12. These plots are consistent with the aggregated results of Table 7.18. A clear distinction between the BRKGA models and the Scatter Search models can be noted. BRKGA+LS produced a superior interquartile range compared to BRKGA, however, the negative outliers of BRKGA+LS are the worst solutions of the entire experiment. The boxplots of SS and SS+LS reveal the same characteristics of Table 7.18, the performance of both models is similar but slightly in favor of SS+LS.

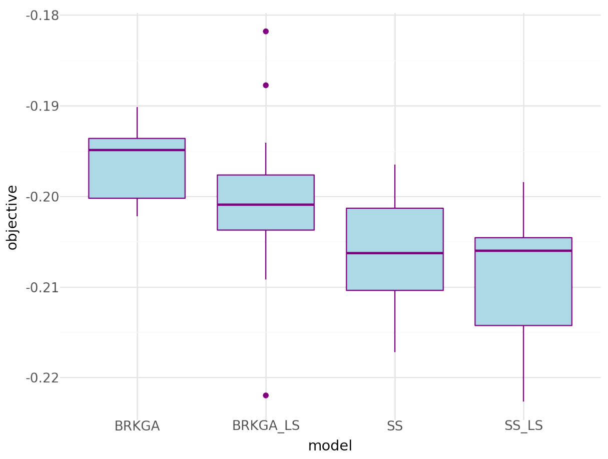

Figure 7.12: Boxplots of objective values of the models of instance size 24

Based on the results in Table 7.18 and Figure 7.12 it is expected that the statistical tests will result in no statistically significant differences in mean objective value between the base models and the local search variants. To verify this, the first step is to perform Shapiro-Wilk tests. These tests concluded that all models produced normal distributed objective values. The subsequent t-tests produced a p-value of 0.0971 for BRKGA and 0.2959 for SS. Both p-values exceed the significance threshold of 0.05, though it was close for BRKGA, which translates to insufficient evidence to claim a difference in mean objective value between the base models and their local search variant.

7.4.5 Conclusion Instance Size Experiment

In summary, the statistical analysis reveals that implementing local search does not produce statistically significant improvements over the base models. Scatter search models consistently outperformed BRKGA models across all instance sizes, suggesting that SS is the more suitable approach to solving the C2Pa problem. Although local search did not show statistical significance, it is worth noting that SS+LS achieved minor improvements in average objective values for instance size 3. SS+LS demonstrated notably lower variability in objective values compared to SS. However, this advantage diminishes as the size of the instance increases. The effectiveness of SS is most apparent in the additional experiment of instance size 6, where SS revealed superior solution quality when using equal computational limits. For instance size 12, when the problem has become more complex, the base SS model emerges as the superior model. For instance size 24, the advantage has flipped to SS+LS. The minimal difference in the average objective values between SS and SS+LS indicates consistently high-quality solutions. As the instance size increases, several patterns emerge in the generated solutions: the objective value deteriorates with more projects, mismatched skills remain relatively stable until instance 24, satisfaction scores steadily decline, and hourly costs remain stable. These patterns, especially in SS, suggest that the model prioritizes skill matching at the expense of satisfaction scores. It is important to note that these results are influenced by the specific projects included in each experiment. For an instance of size 24, the SS model starts showing signs of lower quality assignments. By expanding the portfolio size, the model is instructed to schedule more projects while the workforce remains the same. This results in more restrictive assignments as the projects are assigned, which in the end forces the model to compromise on assignment quality.

7.4.6 Analysis Convergence Speed

The convergence speed of the SS algorithm was analyzed to understand the relationship between the computation time and the number of projects in the portfolio. Polynomial regression was applied to determine this relationship, with the results shown in Figure 7.13. The analysis revealed a relationship of $T = 3.4159 \cdot P^{2.0741}$, where T is the convergence time in seconds and P is the number of projects. With an R^2 value of 0.9911, this model provides an excellent fit to the data. The exponent of approximately 2.07 indicates that the computational time to convergence grows marginally faster than quadratic time as the portfolio size increases.

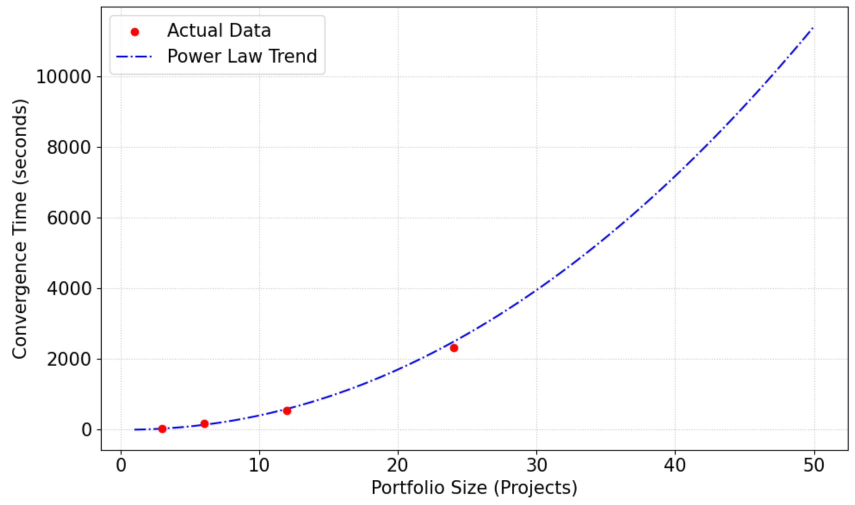

Figure 7.13: Polynomial Regression Fit on Convergence Time SS

7.5 Experiment Flexible Time Windows

This section examines how expanding the flexibility of project start time windows influences the overall solution. The results of this experiment will provide the answer to the second additional research question and will be of added value for answering the main research question. The experiment analyzes various scenarios by implementing time window sizes of 0, 1, 2, 4, and 8 weeks, replacing the original start time windows. Through this experiment, BE NL can better understand the potential advantages of implementing flexible start time windows for their clients.

The experiment utilizes a problem instance of instance size 3. The projects have been chosen in such a way as to represent three service lines where multiple consultants are needed per project. The earliest start times of the projects are week 6, week 9, and week 2. The hypothesis is that broader start time windows should yield better solutions, as the increased flexibility in project start time potentially allows access to a larger pool of available consultants. We anticipate that expanding the time window will result in reduced objective values, skill matches approaching zero more closely, enhanced satisfaction scores, and lower hourly costs.

Although Section 7.4.1 revealed that local search integration did not yield statistically significant improvements for instances of size 3, the local search models, particularly SS+LS, demonstrated more concentrated and on average superior solutions. Furthermore, the extension of start time windows creates a larger solution space, driven by increases in both consultant availability and potential project start times. Given these considerations, the decision was made to include local search variants in this experiment but limit the used models to SS and SS+LS. To address the increased complexity introduced by extending time windows and ensure proper convergence, the number of generations have been modified. The SS models will initially perform 200 generations, however, from TW=2 onward, the number of generations will stepwise increase by 50 to account for the increased complexity. To maintain result reliability and validity, each experimental scenario will be repeated across 15 independent runs.

The results of the experiment showed that SS+LS produced minimal differences compared to SS, which led to the relocation of these results to Appendix 10.3. The aggregated results of the flexible time window experiment using SS can be found in Table 7.19. It mostly confirms the expectation that increasing window size results in increased skill match and satisfaction and decreasing objective value. However, the hourly cost KPI remains around the same level. This could be explained in two ways: either the consultants of lower job positions are fully booked well into the future or the required skill level of the projects is substantially high so that the benefit that skillful consultants of higher job positions bring to the projects outweighs the increased cost. It is also worth noting that the standard deviation of the objective keeps increasing as the window size increases. This is directly related to the length of the chromosome. In the experiments, the length of the chromosome increased as follows: 169→180→215→325→443. The extended flexibility causes added consultant availability; this makes the problem more complex, which decreases the convergence speed.

Exp.	Objective	Skill match	Satisfaction	Hourly cost
	mean ± std dev	**mean ± std dev**	**mean ± std dev**	**mean ± std dev**
TW = 0	-0.2021 ± 0.0009	-1.5901 ± 0.0568	6.121 ± 0.100	152.84 ± 1.31
TW = 1	-0.2005 ± 0.0028	-1.6370 ± 0.0689	6.087 ± 0.123	151.08 ± 3.50
TW = 2	-0.2404 ± 0.0037	-0.9704 ± 0.1429	7.070 ± 0.135	152.90 ± 3.69
TW = 4	-0.2403 ± 0.0044	-0.9580 ± 0.0752	7.030 ± 0.201	152.50 ± 2.40
TW = 8	-0.2506 ± 0.0084	-0.8000 ± 0.2056	7.264 ± 0.186	150.77 ± 3.91

Table 7.19: SS results experiment flexible time windows

The table reveals three distinct time window groups based on objective value: 0 and 1 weeks, 2 and 4 weeks, and 8 weeks. Adding a single week on top of the starting date provided no improvements and actually complicated the problem by introducing additional consultants that were not suitable for the projects. Expanding the time window to 2 or 4 weeks results in superior solutions. The extended flexibility not only leads to more available consultants, but also leads to more availability of consultants that better reflects the requirements of the projects. Compared to the 0 and 1-week group, this approach yields notable improvements: the objective decreases by 0.04 on average, mismatched skills per project reduce by 0.6, and satisfaction increases by a full point. Within this group, extending the time window from 2 to 4 weeks does not produce clear differences, despite the chromosome length increasing by more than half. The increased complexity is not reflected in the standard deviation of the two experiments. The standard deviation of the objective and satisfaction show minor variability increases, whereas the skill match and hourly cost variability actually decrease. Doubling the time window to 8 weeks further improves solution quality across all metrics. However, from a practical standpoint, requesting two months of flexible start-time from a client is unrealistic and, therefore, considered unfavorable.

The experiment strongly supports the implementation of a two-week flexible start time window as a standard business policy for project staffing. Expanding the time window from 0 or 1 week to 2 weeks yields notable improvements in solution quality, including reduced objective values, fewer skill mismatches, and increased overall satisfaction. Although the 4-week window also shows performance enhancements, it becomes less practical when considering business dynamics. Clients typically seek consulting services to solve urgent problems, a month-long waiting period could risk losing their client and driving them to competitors. By adopting a standardized two-week flexible start time, BE NL can systematically improve skill matching and provide quality projects to clients.

7.6 Experiment Business Rules

This section explores how adding business rules to the model affects the solutions that are generated, which answers the final research question. In total, five business rule experiments will be performed. The business rules applied in these experiments are formulated as: strictly following the recommended project specifications, restricting the service lines and introducing little flexibility in the job positions, only restricting the service lines, only introducing little flexibility in the job positions, and having no restrictions. Currently, the company allows any consultant to be assigned to any project, regardless of whether their skills match or if they work in the right service line. The only factor used to determine if a consultant is suitable for a project is their skill match score. From a modeling perspective, these loose rules create problems. The solution space becomes filled with consultants who are not suitable for specific team roles because their service line and job position do not match. Having these unsuitable consultants in the mix adds unnecessary complexity and slows down the convergence speed of the algorithms. Additionally, each team member role in a project usually comes with specific requirements about the job position needed. These requirements exist to make sure projects get team members with the right level of expertise and experience. The current system's approach of ignoring these important specifications makes it harder to find the best possible team combinations.

A problem instance of 3 projects has been put together to investigate the impact of the rules. Table 7.20 outlines the service lines and job positions needed for each project. These projects are handpicked because they include varying service lines and job positions while requiring multiple consultants per project. The investigation will follow several steps. First, a benchmark experiment will be executed that solves the problem the way we did in earlier experiments. Following this, the service line rule and job position rule will be tested both independently and in combination. The effectiveness of each approach will be measured through average scores and standard deviations across all 10 independent runs. Previous research revealed that a flexible time window of 2 weeks offers the most advantageous outcomes within the business context. The Scatter Search (SS) models consistently outperformed the BRKGA models, and local search did not produce statistically significant improvements over the base variants. As a result, the SS model has been selected to generate results for this experiment. Analysis of convergence curves will reveal how quickly each version achieves quality solutions. Finally, a comparison between the benchmark experiment assignments and the actual project requirements listed in Table 7.20 will determine if the assignments align with project specifications.

Project	1: SL & role	2: SL & role	3: SL & role	4: SL & role
0	P&S: SC	P&S: SMA	-	-
1	C&G: SM	C&G: M	TEC: M	D&A: SMA
2	C&G: SC	C&G: SMA	-	-

Table 7.20: Data description of business rules experiment

7.6.1 STRICT Rules

Table 7.21 shows the results of the experiment in order of less restrictive business rules. Analysis of the STRICT experiment, where project specifications from Table 7.20 must be precisely followed, reveals significant limitations. The resulting chromosome length of 28 indicates severely restricted consultant availability. This strict adherence to service lines and roles leads to poor performance outcomes: two-thirds of projects are declined due to the absence of consultants meeting the exact service line and position requirements. Even for the single accepted project, the skill match falls considerably below the standards achieved in previous experiments. These findings demonstrate that strictly following OTL recommendations proves counterproductive, resulting in declined projects and suboptimal consultant assignments.

Exp.	**Objective**	**Skill match**	**Satisfaction**	**Hourly cost**	**Time (sec)**
	mean ± std dev	**mean ± std dev**	**mean ± std dev**	**mean ± std dev**	**mean ± std dev**
STRICT	4.0534 ± 0.0000	-10.000 ± 0.0000	6.000 ± 0.000	127.78 ± 0.00	27.84 ± 1.62
SL+POS	-0.3450 ± 0.0022	-1.3000 ± 0.1741	6.785 ± 0.099	138.61 ± 4.58	71.16 ± 6.06
SL	-0.3594 ± 0.0041	-0.4037 ± 0.1455	6.834 ± 0.104	143.63 ± 4.53	68.41 ± 4.05
POS	-0.3433 ± 0.0048	-1.4852 ± 0.2259	6.884 ± 0.183	139.07 ± 5.02	71.61 ± 3.59
BASE	-0.3518 ± 0.0073	-0.7481 ± 0.4240	6.751 ± 0.132	144.03 ± 5.14	78.28 ± 3.19

Table 7.21: Results experiment business rules

7.6.2 SL+POS Rules

The second experiment (SL+POS) aims to introduce flexibility to the project specifications of Table 7.20. The research introduces a more flexible approach to job position requirements, allowing consultants one rank above or below the recommended position to be considered for team member roles, while maintaining service line specifications. For instance, a role initially requiring a senior consultant (SC) can now be filled by a consultant (C), senior consultant (SC), or manager (M). This flexibility acknowledges that consultants in adjacent job positions often possess comparable experience and expertise levels, developed through similar project engagements and professional development. The resulting chromosome length of 121 indicates that the problem of unavailability has been resolved. The added availability results in the acceptance of all three projects. This experiment configuration will be used as the benchmark for the remainder of the experiment. Therefore, the other experiments will be compared to the skill match of -1.3000 mismatched skills, the satisfaction score of 6.785, and the hourly cost of 138.61.

7.6.3 SL Rules

The third experiment (SL) does not restrict the job positions of consultants when assigning consultants. Consequently, a team member role can be assigned to every available consultant of the associated service line. Implementing this rule resulted in a chromosome of length 147. By not restricting the allowed job positions to a subset, the average objective value is increased from -0.3450 to -0.3594 and the average mismatched skills is improved by almost 70%. In addition to this, the average satisfaction score increases from 6.785 to 6.834. These benefits are achieved by assigning consultants of higher job position since these possess more skills, as the hourly cost increases from 138.61 to 143.63. The spread of the solutions is comparable to SL+POS. To demonstrate the trade-off that is made, the best solutions of SL+POS and SL have been compared. The only difference occurred in project 2, where a senior consultant (SC) and a senior management analyst (SMA) are recommended by the OTL. The SL+POS experiment appointed a manager (M) and a senior management analyst (SMA), while SL replaced the SMA position with another manager, resulting in increased hourly costs. This assignment should be violated for the SL+POS experiment, as team member role 2 strictly allows only MA, SMA, or C positions. Furthermore, a manager ranks three positions higher than the recommended senior management analyst position. Permitting assignments that deviate significantly from OTL recommendations is not advisable, as these recommendations are carefully formulated based on project requirements and intuition. Such deviations from the intended position levels likely conflict with established business policies and could compromise future project quality when these higher positions levels are needed, while unnecessarily inflating project costs now.

7.6.4 POS Rules

The fourth experiment (POS) does not take the recommended service lines into account when assigning consultants. Consequently, a team member role can be assigned to every available consultant that meets the job position interval as proposed in the second experiment. Implementing this rule results in a chromosome of length 190. The absence of service line restrictions fails to yield any improvements in solution quality. While POS generates solutions with similar average values to SL+POS, it demonstrates twice the variability in results. This increased spread can be attributed to the additional complexity introduced by including consultants from all service lines in the solution space. Adding consultants from non-matching service lines essentially adds noise to the problem, making it more challenging for the algorithm to consistently find quality solutions. Note that a perfect alignment of the required service line and required skills is essential for this to occur. If the OTL fails to deliver perfectly aligned project specifications, this experiment would prove its worth and signal errors in the project data.

7.6.5 No Rules

The fifth and final experiment (BASE) removes all aforementioned restrictions and solves the problem as was done in the previous sections. Having no restrictions in place resulted in a chromosome of length 291, which is significantly larger than the other experiments. In terms of average performance, the quality of the solutions can be placed between SL+POS and SL. However, the objective and skill match metrics show notably higher variability compared to other experiments. Similar to the previous findings, this increased spread in results originates from the larger pool of available consultants included in the solution space.

Figure 7.14 shows the convergence curves of the first iteration of the last four experiments. It can be seen that the experiments where rules are implemented show signs that the problem has converged before the maximum number of generations have been reached. The BASE experiment has not yet converged, since the (close to) optimal solutions should be around -0.36, as demonstrated by the results of SL. So, in addition to generating solutions that are more reliable compared to BASE, the business rule experiments also prove to have converged when the BASE experiment has not converged, indicating that the business rule experiments have an advantage in terms of convergence speed.

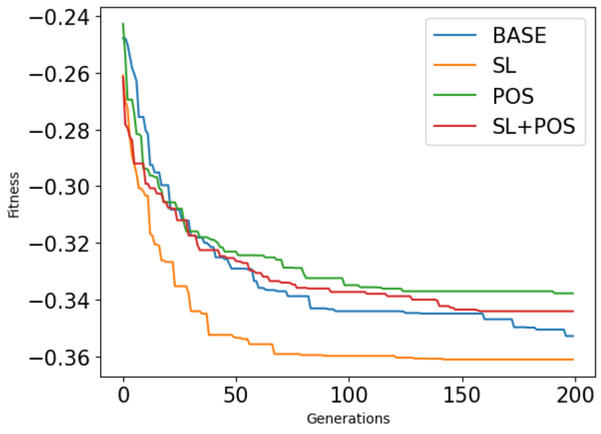

Figure 7.14: Convergence curves business rules experiment

7.6.6 Conclusion Business Rules

In summary, restricting the service lines and the job positions as proposed in SL+POS seems to be most applicable to the C2Pa procedure. This is because it utilizes the thoughtful recommendations from the OTL and adds some flexibility to the job positions to produce solutions that are still in line with the ideas of the OTL. SL takes advantage of the importance of the skill match KPI to assign consultants that are ranked much higher than prescribed, where POS reduces to being a copy of SL+POS with added noise. Having no rules implemented (BASE) does not seem to yield any benefits over SL, whereas strictly following the project specifications forces BE NL to decline projects.

The experimental results show that restricting only service lines (SL) produced the highest quality solutions. This configuration demonstrated robust performance with relatively low standard deviations across all metrics, and showed superior solution quality and convergence speed compared to the BASE configuration. Nevertheless, the SL+POS experiment stands out as the most practical approach, despite not achieving the absolute best numerical results. This implementation, which combines service line restrictions with flexible job position requirements, achieves an effective balance between optimization and business reality. The approach maintains appropriate expertise levels and skill alignment while preserving a reasonable pool of available consultants. By adopting this balanced policy, BE NL can better align its staffing practices with business objectives, acknowledging that optimal solutions may not always translate to the best practical outcomes. The experiment demonstrates that the model can be adjusted to match BE NL's desired business policies, offering a flexible framework that delivers consistent, high-quality staffing solutions while respecting operational constraints.

8 Conclusion

This research aimed to determine the effectiveness of (hybrid-)meta-heuristics on the C2Pa problem. The primary objective was to develop a meta-heuristic model integrated with local search capable of generating high-quality solutions while also prioritizing efficiency. The main research question connected to this objective is: "Does the implementation of local search boost the performance of meta-heuristics in solving the Consultant-to-Project Assignment (C2Pa) problem at BearingPoint NL, specifically focusing on efficiency and efficacy?" To answer this, the developed models were used to perform experiments on instance size, flexible time windows, and business rules. Throughout these experiments, a comparison was made between versions of the models with and without local search.

The experiments yielded several key findings: The Scatter Search (SS) models demonstrated superior solution quality compared to the BRKGA models in their current implementation. The experiments in Section 7.4 showed a clear advantage of SS over BRKGA. Notably, while local search did not produce statistically significant improvements in solution quality over the base models, SS+LS generated marginally better solutions for the smallest instance. However, this advantage of local search disappeared in the instances of sizes 6 and 12. The generated solutions showed the following pattern as the instance size increased: the objective value deteriorated with more projects, mismatched skills remained relatively stable, satisfaction scores steadily declined, hourly costs remained stable, while the client project utilization rate remained insignificant since no projects were declined. Section 7.5 determined that an 8-week flexible start time window produced optimal solutions. The addition of local search did not generate notably improved solution quality compared to those produced by SS alone. Taking both the business context and numerical results into account, a 2-week flexible start time window emerges as the most practical recommendation. This proposed time window is expected to enhance both average mismatched skills and average satisfaction, while maintaining stable average hourly costs. Finally, Section 7.6 indicated that configurations restricting only service lines generated the best solutions. However, this approach significantly deviates from the recommendations of the OTL. Therefore, the recommended configuration involves restricting the service line while maintaining little flexibility in allowed job positions, as this approach aligns closely with OTL recommendations while consistently producing robust, high-quality solutions.

Based on the findings, the main research question can be addressed as follows: The research did not find evidence supporting the conclusion that the implementation of local search boosts the efficacy of meta-heuristics in solving the C2Pa problem. The introduction of local search techniques not only failed to produce statistically significant improvements but also substantially impacted the models' efficiency. Specifically, the experiments demonstrated that the implementation of local search led to a decrease in efficiency ranging from 68.00% to 181.69%. Consequently, the base variant of Scatter Search should be used over its local search counterpart as it achieves similar solution quality while not having the disadvantages that come with local search.

The contributions of this research reach beyond the implementation of a meta-heuristic with local search on the C2Pa problem. Firstly, an encoder was developed that translated the entire problem into a chromosome using the Random Key representation that includes project ordering, available consultants, involved skills, and time windows. The developed decoder translates the chromosome into a guaranteed feasible solution. The research gap regarding project selection was solved using the priority system based on the real number ordering of the projects. In addition to that, a scheduling and staffing model is developed that does not rely on precedence relations between the projects and team member roles. Lastly, the proposed simultaneous scheduling and staffing models were driven by a multi-objective fitness function that was not related to the starting or finishing times of projects. With that, all the research gaps are solved.

This research provides a meta-heuristic framework that can be used to solve similar simultaneous scheduling and staffing problems. An extensive comparison between two solution approaches has been executed, demonstrating that the Scatter Search methodology is a promising option to use as a meta-heuristic for these kind of problems. Experiments revealed that the framework can easily be adjusted to changing requirements. Furthermore, the demonstrated scalability of the model suggests its applicability extends beyond the scale of BearingPoint NL's operations to larger practices. This research provided BearingPoint NL with a promising model that can be utilized as a decision support tool within the OTL team. In addition to that, evidence-based recommendations are generated that could improve the business policy of the OTL. Further research could be directed towards additional algorithmic features that increase the exploration capability of Scatter Search or algorithmic features that increase the exploitation capability of BRKGA. Lastly, more research could be done to investigate the potential benefits of local search after implementing the recommendations.

9

Discussion

This chapter discusses the limitations of the research and hints at possible improvements that can be made in future research and implementation. This includes data requirements, modeling details, and the underlying rules.

The practical applicability of the proposed solution approach heavily depends on data completeness and quality. During this research, two significant data-related challenges emerged: consultants with incomplete skillset data (resulting in skill levels of 0 and satisfaction scores of 1) and consultants with entirely missing data. Both scenarios led to the model excluding these consultants from the assignment process. For the model to be effectively implemented in practice, BE NL must ensure that consultant skill data is not only complete but also maintained through regular updates. Moreover, regular checks for validity should be performed by consultants of high job position as the answers to the survey can be manipulated.

The current feasibility check of the model may be overly restrictive, as it blocks the availability of a consultant for an entire project if there is insufficient time in just one week of the project duration. A more nuanced approach could be implemented by introducing a small margin of flexibility in the feasibility check. This would better reflect real-world practices, where consultants are occasionally permitted to work slightly beyond their contractual hours or could take a few days of during a project. Such flexibility could expand the solution space and potentially lead to better overall assignments.

The scope of this research was to implement two promising meta-heuristics and demonstrate the effectiveness of the base variants and the local search variants by means of the experiments. More research could be done on testing other promising approaches and improving the proposed models. For instance, the BRKGA model could be enhanced by incorporating methods that increase selection pressure in the recombination step, such as tournament selection for choosing the second parent. Implementation of such methods could potentially improve the model's exploitation capabilities. Furthermore, additional advanced methods could be integrated into the Scatter Search model, e.g. as shown in the paper of Marti et al. (35). These methods could enhance the model's ability to escape local optima, thereby improving its exploration capabilities.

The fact that skills are not strictly bound to specific team member roles allows for exploits. The set of skills in this problem can be categorized into two groups: generic consultant skills and service line-specific skills. By not restricting the involved project skills to certain team members, a consultant who lacks the service line-specific skills (since they are from a different service line) could still be assigned to the project, as they can be responsible for the generic consultant skills. However, the OTL team would likely avoid such an assignment, as every consultant should be partially responsible for the generic consultant skills. This issue can be addressed by linking certain skills to predetermined project team member roles. In addition, the number of skills assigned to each consultant is determined by the required work hours, making the assumption that each skill requires an equal amount of time. However, in reality, the time demands of different skills can vary significantly. A complete linking of required skills to team member roles could potentially resolve this issue. Such an approach would eliminate the skill division problem, thus reducing the overall complexity. Alternatively, the model could allow certain skills to remain unrestricted or restrict them to be assigned to consultants from specific service lines, which would preserve the skill division problem. It is clear that the skill division aspect of the problem requires further refinement to better align with real-world conditions.

The limited effectiveness of especially local search consultant swap and the overall low convergence ability stem from the complex solution space. For consultant swap to generate an improvement, the skills should be perfectly aligned for the trial consultant to result in an improvement. Whenever a solution enters local search, the quality of the solution is already adequate. Considering that projects can require multiple consultants that all include multiple skills, the requirements needed for a trial consultant to be a better fit on average compared to the assigned consultant are quite steep. In addition to that, applying small perturbations to a solution could yield substantial changes in terms of fitness. The complex interaction between consultant assignments and skill division in multi-consultant projects limits the convergence ability of the algorithms. Any refinement to the skill division aspect of the problem should increase the overall convergence ability of the algorithms and the effectiveness of local search.

The computational demands of calculating the KPIs of the fitness function significantly limited the frequency and scope of local search. With the implementation of business rules on consultant availability, which effectively reduces the problem size, further research should investigate whether the use of local search could lead to improvements in solution quality. This consideration also applies to the recommended adjustments in the skill division aspect of the problem. Additionally, since the entire problem was coded from scratch in Python, optimizing code efficiency could potentially allow for more frequent application of local search techniques.

Further research could explore alternative approaches to KPI weighting. The current method, using point allocation, is primarily based on intuitive judgments. Future research could investigate more theoretically grounded weighting methodologies. Such methodologies could provide a more robust foundation for balancing the various performance indicators in the optimization process.

References

[1] B. N. Zentveld. *Improving BearingPoint Netherlands' consultant-to-project KPIs by introducing a consultant-to-project assignment model.* Master's thesis, University of Twente, 2024. 1, 3, 4, 5, 7, 11, 27, 29, 40, 68, 104

[2] S. H. Dike. **Project scheduling with resource constraints**. *IEEE Transactions on Engineering Management*, (4):155–157, 1964. 11

[3] J. K. Lenstra J. Blazewicz and A. R. Kan. **Scheduling subject to resource constraints: Classification and complexity**. *Discrete Applied Mathematics*, **5**(1):11–24, 1983. 11

[4] S. Hartman and D. Briskorn. **A survey of variants and extensions of the resource-constrained project scheduling problem**. *European Journal of Operational Research*, **207**(1):1–14, 2010. 11

[5] E. W. Davis. *An exact algorithm for the multiple constrained-resource project scheduling problem.* Yale University, 1968. 12

[6] C. Zhuang H. Ding and J. Liu. **Extensions of the resource-constrained project scheduling problem**. *Automation in Construction*, **153**:104958, 2023. 12

[7] B. Afshar-Nadjafi. **Multi-skilling in scheduling problems: A review on models, methods and applications**. *Computers Industrial Engineering*, **151**:107004, 2021. 12

[8] C. Heimerl and R. Kolisch. **Scheduling and staffing multiple projects with a multi-skilled workforce**. *OR Spectrum*, **32**:343–368, 2010. 12

[9] J. L. Zhu J. J. Chen and D. N. Zhang. **Multi-project scheduling problem with human resources based on dynamic programming and staff time coefficient**. In *2014 International Conference on Management Science Engineering 21th Annual Conference Proceedings*, pages 1012–1018. IEEE, 2014. 13

[10] N. Perrier R. Pellerin and F. Berthaut. **A survey of hybrid metaheuristics for the resource-constrained project scheduling problem**. *European Journal of Operational Research*, **280**(2):395–416, 2020. 13

[11] R. Kolisch and A. Sprecher. **PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program**. *European Journal of Operational Research*, **96**(1):205–216, 1997. 13, 16

[12] D. Debels and M. Vanhoucke. **A bi-population based genetic algorithm for the resource-constrained project scheduling problem**. In *International Conference on Computational Science and Its Applications*, pages 378–387. Springer Berlin Heidelberg, 2005. 13

[13] C.D. Tarantilis D.C. Paraskevopoulos and G. Ioannou. **Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm**. *Expert Systems with Applications*, **39**(4):3983–3994, 2012. 13, 14

[14] I. Correia B. F. Almeida and F. Saldanha da Gama. **A biased random-key genetic algorithm for the project scheduling problem with flexible resources**. *Top*, **26**(2):283–308, 2018. 14, 15

[15] J.J. Mendes J.F. Gonçalves and M.G. Resende. **A genetic algorithm for the resource constrained multi-project scheduling problem**. *European Journal of Operational Research*, **189**(3):1171–1190, 2008. 15, 16, 50

[16] D. Bredael and M. Vanhoucke. **A genetic algorithm with resource buffers for the resource-constrained multi-project scheduling problem**. *European Journal of Operational Research*, **315**(1):19–34, 2024. 15, 16

[17] V. Van Peteghem and M. Vanhoucke. **An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances**. *European Journal of Operational Research*, **235**(1):62–72, 2014. 16, 17

[18] H. Maghsoudlou, B. Afshar-Nadjafi, and S. T. A. Niaki. **Multi-skilled project scheduling with level-dependent rework risk; three multi-objective mechanisms based on cuckoo search**. *Applied Soft Computing*, **54**:46–61, 2017. 16, 17, 20

[19] L. F. Machado-Domínguez, C. D. Paternina-Arboleda, J. I. Vélez, and A. Barrios-Sarmiento. **A memetic algorithm to address the multi-node resource-constrained project scheduling problem**. *Journal of Scheduling*, **24**:413–429, 2021. 17

[20] V. Van Peteghem and M. Vanhoucke. **Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem**. *Journal of Heuristics*, **17**:705–728, 2011. 17, 56, 58, 59

[21] D. Gu R. Chen, C. Liang and J.Y. Leung. **A multi-objective model for multi-project scheduling and multi-skilled staff assignment for IT product development considering competency evolution**. *International Journal of Production Research*, **55**(21):6207–6234, 2017. 18

[22] C. Yugma C. Gallais R. Torba, S. Dauzère-Pérès and J. Pouzet. **Solving a real-life multi-skill resource-constrained multi-project scheduling problem**. *Annals of Operations Research*, pages 1–46, 2024. 18, 19

[23] J. C. Bean. **Genetic algorithms and random keys for sequencing and optimization**. *ORSA Journal on Computing*, **6**(2):154–160, 1994. 30, 49

[24] J. F. Gonçalves and M. G. Resende. **Biased random-key genetic algorithms for combinatorial optimization**. *Journal of Heuristics*, **17**(5):487–525, 2011. 31, 49

[25] R. Kolisch and S. Hartmann. *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis.* Springer US, 1999. 34

[26] F. H. Barron and B. E. Barrett. **Decision quality using ranked attribute weights**. *Management Science*, **42**(11):1515–1523, 1996. 41

[27] P. A. Bottomley and J. R. Doyle. **A comparison of three weight elicitation methods: good, better, and best**. *Omega*, **29**(6):553–560, 2001. 42

[28] S.K. Lam, A. Pitrou, and S. Seibert. **Numba: A LLVM-based Python JIT Compiler**. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015. 44

[29] A.A. Chaves, J.F. Gonçalves, and L.A.N. Lorena. **Adaptive Biased Random-Key Genetic Algorithm with Local Search for the Capacitated Centered Clustering Problem**. *Computers & Industrial Engineering*, **124**:331–346, 2018. 50

[30] C.E. Andrade, T. Silva, and L.S. Pessoa. **Minimizing Flowtime in a Flowshop Scheduling Problem with a Biased Random-Key Genetic Algorithm**. *Expert Systems with Applications*, **128**:67–80, 2019. 51, 54

[31] H. Ishibuchi, T. Yoshida, and T. Murata. **Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling**. *IEEE Transactions on Evolutionary Computation*, **7**(2):204–223, 2003. 51

[32] A. E. Eiben and S. K. Smit. **Parameter tuning for configuring and analyzing evolutionary algorithms**. *Swarm and Evolutionary Computation*, **1**(1):19–31, 2011. 53

[33] S. K. Karna and R. Sahai. **An overview on Taguchi method**. *International Journal of Engineering and Mathematical Sciences*, **1**(1):1–7, 2012. 54

[34] F. Glover. **Heuristics for Integer Programming Using Surrogate Constraints**. *Decision Sciences*, **8**(1):156–166, 1977. 56

[35] R. Martí, M. Laguna, and F. Glover. **Principles of Scatter Search**. *European Journal of Operational Research*, **169**(2):359–372, 2006. 56, 97

[36] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2015. 71

[37] F. J. Lobo, C. F. Lima, and Z. Michalewicz. *Parameter Setting in Evolutionary Algorithms*, **54**. Springer Science & Business Media, 2007. 72

[38] A. Alorf. **A survey of recently developed metaheuristics and their comparative analysis**. *Engineering Applications of Artificial Intelligence*, **117**:105622, 2023. 111

[39] O. Bozorg-Haddad I. Ahmadianfar and X. Chu. **Gradient-based optimizer: A new metaheuristic optimization algorithm**. *Information Sciences*, **540**:131–159, 2020. 111

[40] M. Y. Cheng and D. Prayogo. **Symbiotic organisms search: a new metaheuristic optimization algorithm**. *Computers & Structures*, **139**:98–112, 2014. 112

[41] V. A. Hodianto and I. Yang. **Multi-Mode Resource Constrained Multi-Project Scheduling Problem Optimization with Symbiotic Organisms Search**. *Dimensi Utama Teknik Sipil*, **9**(1):77–96, 2022. 112, 113

10

Appendix

10.1 MILP model

Below, the mathematical model and the list of parameters and variables of the MILP model of (1) can be found.

4.2 Model description

We can now model the following mixed integer linear programming (MILP) model:

Indices	**Description**
i	Consultant index i with $i = 1, \dots, I$
m	Team member index m with $m = 1, \dots, m_p$
p	Project index p with $p = 1, \dots, P$
s	Skill index s with $s = 1, \dots, S$
t	Time index t with $t = 1, \dots, T$

Parameters	**Description**
α^+_{cost} and α^-_{cost}	Positive and negative weight attached to the normalized sum of deviations of e^+_{cost} and e^-_{cost}
α^+_{util} and α^-_{util}	Positive and negative weight attached to the normalized sum of deviations of e^+_{util} and e^-_{util}
α^+_{satis} and α^-_{satis}	Positive and negative weight attached to the normalized sum of deviations of e^+_{satis} and e^-_{satis}
c_i	Hourly cost parameter for assigning consulant i to a project based on the consultant's job position (see Table 2.1)
$CH_{t,p,s}$	Chargeable hours parameter for required skill s of project p at period t
$csl_{s,i}$	Consultant i's skill level parameter for skill s (according to the skill level range as specified in Table B.1)
$css_{s,i}$	Consultant i's skill satisfaction parameter for skill s (according to the satisfaction scale as specified in Table B.2)
$em_{i,p}$	Binary parameter that equals 1 in case consultant i is working as EM on project p and 0 otherwise.
emh	Parameter that represents the hours an EM is working as EM per project per week.
ES_p	Project p's earliest start time parameter
I	Number of consultants parameter

Parameters	**Description**
LF_p	Project p's latest finish time where $LF_p = LS_p + pt_p - 1$
LS_p	Project p's latest start time parameter
m_p	Number of team members in project p parameter
$NCH_{t,p,s}$	Non-chargeable hours parameter for required skill s of project p at period t
$NH_{i,t}$	Net hours parameter of consultant i in period t
P	Number of projects parameter
$PH_{p,m}$	Number of project hours for team member m of project p for every t during the project. $PH_{p,m}$ remains constant during the project (does not change).
$psl_{s,p}$	Project p's skill requirement level parameter for skill s (according to the skill levels range as specified in Table B.1)
pt_p	Project p's processing time parameter
pv_p	Priority value parameter for project p according to (Subsection 2.4.3)
$q_{p,s,i}$	Binary parameter underqualification for skill s for consultant i on project p. $q_{p,s,i}$ = $\begin{cases} 1\, if\, psl_{s,p} \geq csl_{s,i} \\ 0\, otherwise \end{cases}$
$rs_{p,s}$	Binary parameter if skill s is required for project p. $rs_{p,s}$ = $\begin{cases} 1\, if\, psl_{s,p} \geq 1 \\ 0\, otherwise \end{cases}$
S	Number of skills parameter
$st_{p,m}$	Number of required skills of project p that are assigned to team member m parameter.
T	Number of time in planning horizon parameter
$TotMaxAbsCost$	The total maximum absolute hourly cost is a parameter that represents the maximum absolute value of the hourly cost of assigned consultants KPI. The maximum absolute cost value is 165.
$TotMaxAbsMatch$	The total maximum absolute skill match is a parameter that represents the maximum absolute value of the C2Pa skill match KPI. The maximum absolute C2Pa skill match value is reached when a project requires a skill at expert level (3) while the consultant does not have this skill (0). The skill match for the required project skill would then become $0^2 - 3^2 = 9$. Since in the absolute worst case this could be required for all project skills and all projects we multiply this value by the number of projects P and the number of skills S. Therefore $TotMaxAbsMatch = (\max(PSL_{s,p}))^2 \cdot P \cdot S$.
$TotMaxAbsSatis$	The total maximum absolute satisfaction is a parameter that represents the maximum absolute value of the consultant satisfaction KPI. The maximum satisfaction that a consultant can give a certain skill is 10. Therefore $TotMaxAbsSatis = 10$.
$TotMaxAbsUtil$	The total maximum absolute consultant-to-project utilization is a parameter that represents the maximum absolute value of the consultant-to-project utilization KPI. The maximum utilization that a consultant can achieve is 100 percent. Therefore $TotMaxAbsSatis = 100$.
TPT_{max}	Parameter that represents the total project hours for all projects in the planning horizon.
TS_{max}	Sum of the number of required skills parameter for all projects in the planning horizon.

Parameters	Description
$w_{p,s,i}$	Binary parameter overqualification for skill s for consultant i on project p. $w_{p,s,i} = \begin{cases} 1\, if\ csl_{s,i} \geq psl_{s,p} \\ 0\ otherwise \end{cases}$
$w_{decline}$	Weight attached to the normalized penalty factor for declining a project
w^{+}_{cost} and w^{-}_{cost}	Positive and negative weight attached to normalized deviations of the hourly cost of assigned consultants goal (d^{+}_{cost} and d^{-}_{cost})
w^{+}_{util} and w^{-}_{util}	Positive and negative weight attached to normalized deviations of the total consultant-to-project utilization rates goal (d^{+}_{util} and d^{-}_{util})
w^{+}_{satis} and w^{-}_{satis}	Positive and negative weight attached to normalized deviations of the consultant satisfaction rates goal (d^{+}_{satis} and d^{-}_{satis})
w^{+}_{match} and w^{-}_{match}	Positive and negative weight attached to normalzied deviations of the C2Pa skill match goal (d^{+}_{match} and d^{-}_{match})
$WH_{i,t}$	Parameter that represents the working hours of consultant i at time t

Decision variables	Description
$u_{p,t}$	Binary decision variable that equals 1 if project p starts at time t and 0 otherwise.
$x_{i,p,t,m,s}$	Binary decision variable that equals 1 in case consultant i is assigned to team member slot m and skill s of project p at time t, and 0 otherwise.
ap_p	Binary decision variable that equals 1 in case project p is accepted (all team member slots are assigned to consultants), and 0 otherwise.

Auxiliary variables	Description
d^{+}_{cost} and d^{-}_{cost}	Positive and negative deviation variable between the hourly cost of assigned consultants KPI and the continuous variable that represents a value in the aspiration level interval range
$d^{+}_{i,util}$ and $d^{-}_{i,util}$	Positive and negative deviation variable between the consultant-to-project utilization rates KPI and the continuous variable that represents a value in the aspiration level interval range for consultant i
$d^{+}_{i,p,match}$ and $d^{-}_{i,p,match}$	Positive and negative deviation variable between the C2Pa skill match KPI and the continuous variable that represents a value in the aspiration level interval range for consultant i and project p
d^{+}_{satis} and d^{-}_{satis}	Positive and negative deviation variable between the consultant satisfaction rates KPI and the continuous variable that represents a value in the aspiration level interval range
e^{+}_{cost} and e^{-}_{cost}	Positive and negative deviation variable between the continuous variable that represents a value in the aspiration level interval range and the desired minimum or maximum aspiration level of the hourly cost of assigned consultants KPI

Auxiliary variables	**Description**
$e^+_{i,util}$ and $e^-_{i,util}$	Positive and negative deviation variable between the continuous variable that represents a value in the aspiration level interval range and the desired minimum or maximum aspiration level of the consultant-to-project utilization rates KPI for consultant i
e^+_{satis} and e^-_{satis}	Positive and negative deviation variable between the continuous variable that represents a value in the aspiration level interval range and the desired minimum or maximum aspiration level of the consultant satisfaction rates KPI
$pd_{p,t}$	Binary auxiliary variable that equals 1 in case project p is in execution at time t and 0 otherwise
TPT	Auxiliary variable that represents the total project hours for all accepted projects in the planning horizon.
TS	Auxiliary variable that represents the number of required skills parameter for all accepted projects in the planning horizon.
$v_{i,p,m,s}$	Binary auxiliary variable that equals 1 in case consultant i is assigned to team member role m and skill s of project p and 0 otherwise
y_{cost}	Continuous variable that represents a value in the aspiration level interval range of the hourly cost of assigned consultants KPI
$y_{i,util}$	Continuous variable that represents a value in the aspiration level interval range of the consultant-to-project utilization rates KPI for consultant i
y_{satis}	Continuous variable that represents a value in the aspiration level interval range of the consultant satisfaction rates KPI
$z_{i,p,m}$	Binary auxiliary variable that equals 1 in case consultant i is working as team member m on project p

Minimize

$$w^+_{cost} \cdot \frac{d^+_{cost}}{TotMaxAbsCost} + w^-_{cost} \cdot \frac{d^-_{cost}}{TotMaxAbsCost} + \alpha^+_{cost} \cdot \frac{e^+_{cost}}{TotMaxAbsCost} + \alpha^-_{cost} \cdot \frac{e^-_{cost}}{TotMaxAbsCost}$$
$$+\frac{1}{I} \cdot \sum_{i=1}^{I} \left(w^+_{util} \cdot \frac{d^+_{i,util}}{TotMaxAbsUtil} + w^-_{util} \cdot \frac{d^-_{i,util}}{TotMaxAbsUtil} + \alpha^+_{util} \cdot \frac{e^+_{i,util}}{TotMaxAbsUtil} + \alpha^-_{util} \cdot \frac{e^-_{i,util}}{TotMaxAbsUtil} \right)$$
$$+w^+_{satis} \cdot \frac{d^+_{satis}}{TotMaxAbsSatis} + w^-_{satis} \cdot \frac{d^-_{satis}}{TotMaxAbsSatis} + \alpha^+_{satis} \cdot \frac{e^+_{satis}}{TotMaxAbsSatis} + \alpha^-_{satis} \cdot \frac{e^-_{satis}}{TotMaxAbsSatis}$$
$$+\frac{1}{P} \cdot \frac{1}{I} \cdot \sum_{p=1}^{P} \sum_{i=1}^{I} \left(w^+_{match} \cdot \frac{d^+_{i,p,match}}{TotMaxAbsMatch \cdot pt_p} + w^-_{match} \cdot \frac{d^-_{i,p,match}}{TotMaxAbsMatch \cdot pt_p} \right)$$
$$+\frac{1}{P} \cdot \sum_{p=1}^{P} w_{decline} \cdot pv_p \cdot (1 - ap_p) \tag{4.35}$$

Subject to:

$$\frac{\sum_{i=1}^{I} \sum_{p=1}^{P} \sum_{t=ES_p}^{LF_p} \sum_{m=1}^{m_p} \sum_{s=1}^{S} c_i \cdot x_{i,p,t,m,s}}{TPT} - d^+_{cost} + d^-_{cost} = y_{cost} \tag{4.36}$$

$$\frac{\sum_{p=1}^{P} \sum_{t=0}^{T} (\sum_{m=1}^{m_p} \sum_{s=1}^{S} CH_{t,p,s} \cdot x_{i,p,t,m,s}) + WH_{i,t} - NH_{i,t} + emh \cdot em_{i,p} \cdot pd_{p,t}}{(\sum_{t=1}^{T} WH_{i,t}) \cdot T} \cdot 100$$
$$-d^+_{i,util} + d^-_{i,util} = y_{i,util} \quad \forall i \tag{4.37}$$

$$\frac{\sum_{i=1}^{I}\sum_{p=1}^{P}\sum_{t=ES_p}^{LF_p}\sum_{m=1}^{m_p}\sum_{s=1}^{S} x_{i,p,t,m,s}\cdot css_{s,i}}{TS} - d^{+}_{satis} + d^{-}_{satis} = y_{satis} \tag{4.38}$$

$$\sum_{t=ES_p}^{LF_p}\sum_{m=1}^{m_p}(\sum_{s=1}^{S}(csl^2_{s,i} - psl^2_{s,p})\cdot q_{p,s,i} + (csl_{s,i} - psl_{s,p})\cdot w_{p,s,i}) \cdot pv_p \cdot x_{i,p,t,m,s} - d^{+}_{i,p,match} + d^{-}_{i,p,match} = 0 \quad \forall i,p \tag{4.39}$$

$$\sum_{p=1}^{P}\sum_{m=1}^{m_p}\sum_{s=1}^{S}\left(CH_{t,p,s}\cdot x_{i,p,t,m,s} + \frac{emh\cdot em_{i,p}\cdot pd_{p,t}}{m_p\cdot S} + NCH_{t,p,s}\cdot x_{i,p,t,m,s}\right) \leq NH_{i,t} \quad \forall i,t \tag{4.40}$$

$$x_{i,p,t,m,s} \leq ap_p \quad \forall i,p,t,m,s \tag{4.41}$$

$$\sum_{i=1}^{I} z_{i,p,m} \leq 1 \quad \forall p,m \tag{4.42}$$

$$\sum_{m=1}^{m_p} z_{i,p,m} \leq 1 \quad \forall p,i \tag{4.43}$$

$$x_{i,p,t,m,s} \leq z_{i,p,m} \quad \forall i,p,t,m,s \tag{4.44}$$

$$\sum_{i=1}^{I}\sum_{m=1}^{m_p} x_{i,p,t,m,s} \leq 1 \quad \forall p,t,s \tag{4.45}$$

$$\sum_{t=ES_p}^{LS_p} u_{p,t} = ap_p \quad \forall p \tag{4.46}$$

$$x_{i,p,t,m,s} \leq 0 \quad t \notin \{ES_p, ..., LF_p\} \text{ and } \forall i,p,m,s \tag{4.47}$$

$$x_{i,p,t,m,s} \leq rs_{s,p} \quad \forall i,p,t,m,s \tag{4.48}$$

$$\sum_{i=1}^{I}\sum_{s=1}^{S} x_{i,p,t,m,s} = st_{p,m}\cdot pd_{p,t} \quad \forall p,t,m \tag{4.49}$$

$$pt_p\cdot u_{p,t} \leq \sum_{t=t}^{min(t+pt_p,T)} pd_{p,t} \quad t \in \{ES_p, ..., LS_p\} \text{ and } \forall i,p,m,s \tag{4.50}$$

$$\sum_{t=1}^{T} pd_{p,t} \leq pt_p\cdot ap_p \quad \forall p \tag{4.51}$$

$$\sum_{i=1}^{I}\sum_{m=1}^{m_p} v_{i,p,m,s} = ap_p \quad \forall p,s \tag{4.52}$$

$$x_{i,p,t,m,s} = v_{i,p,m,s} \quad \forall i,p,t,m,s \tag{4.53}$$

$$y_{cost} - e^{+}_{cost} + e^{-}_{cost} = 0 \tag{4.54}$$

$$y_{i,util} - e^{+}_{i,util} + e^{-}_{i,util} = 100 \quad \forall i \tag{4.55}$$

$$y_{satis} - e^{+}_{satis} + e^{-}_{satis} = 10 \tag{4.56}$$

$$x_{i,p,t,m} \in \{0,1\} \tag{4.57}$$

$$u_{p,t} \in \{0,1\} \tag{4.58}$$

$$ap_{p} \in \{0,1\} \tag{4.59}$$

$$pd_{p,t} \in \{0,1\} \tag{4.60}$$

$$v_{i,p,m,s} \in \{0,1\} \tag{4.61}$$

$$z_{i,p,m} \in \{0,1\} \tag{4.62}$$

$$TPT \geq 0 \tag{4.63}$$

$$TS \geq 0 \tag{4.64}$$

$$d^{+}_{k}, d^{-}_{k}, e^{+}_{k}, e^{-}_{k} \geq 0 \tag{4.65}$$

$$115 \leq y_{cost} \leq 127 \tag{4.66}$$

$$\begin{gathered} 90 \leq y_{i,util} \leq 100 \quad \text{(in case consultant } i \text{ is } MA) \\ 80 \leq y_{i,util} \leq 100 \quad \text{(in case consultant } i \text{ is } C) \\ 75 \leq y_{i,util} \leq 100 \quad \text{(in case consultant } i \text{ is } SC) \\ 65 \leq y_{i,util} \leq 100 \quad \text{(in case consultant } i \text{ is } M) \end{gathered} \tag{4.67}$$

$$6 \leq y_{satis} \leq 10 \tag{4.68}$$

10.2 Unsuccessful Models

This section focuses on models that are highly regarded in the literature for their promising performance in general numerical optimization problems or C2Pa-related optimization problems. However, their performance on C2Pa was not sufficiently convincing to warrant further investigation or implementation. These experiments were carried out in the early stages of the research, some of the definitions could differ from the main research.

10.2.1 Gradient-Based Optimization

The gradient-based optimization algorithm (GBO) was implemented and modified for the C2Pa problem. Alorf (38) investigated the optimization capability of 26 recently published novel meta-heuristics. The performance of these meta-heuristics is assessed by evaluating 79 benchmarks covering unimodal, multimodal, CEC-BC-2017 benchmarks and constrained engineering problems. These benchmarks mostly cover complex mathematical functions or constrained practical problems. GBO was found to be among the top five best performs for multimodal functions. The impressive results in the benchmarks were convincing enough to consider this algorithm. This algorithm also allows for a comparison between general optimization algorithms and domain specific algorithms.

Ahmadianfar (39) proposed GBO in his study. The Gradient-Based Optimizer (GBO) draws inspiration from Newton's gradient-based method and relies on two primary mechanisms: the Gradient Search Rule (GSR) and the Local Escaping Operator (LEO). It also utilizes a group of vectors to investigate the solution space. The GSR implements a gradient-based approach to improve exploratory capabilities and speed up convergence, leading to more optimal positions within the search area. Meanwhile, the LEO is designed to help the GBO avoid getting trapped in local optima, allowing it to continue searching for better solutions. The MATLAB source code of GBO is made publicly available by Ahmadianfar (39), consequently, this code is succesfully converted into Python code. The functions are adapted to the C2Pa problem. The algorithm makes use of the random key representation, the same representation as the proposed models. Consequently, the same decoding procedure has been applied.

A validation session of iterative simulations on the small instance of 15 consultants and 9 projects conclusively demonstrated that further investment of resources into improving this meta-heuristic would be unproductive. The algorithm's capability to find reasonable solutions proved markedly insufficient, as evidenced by the fitness curves of the best fitness and average population fitness depicted in Figure 10.1. GBO shows significant deficiencies in both exploitation and exploration aspects: Exploitation: The best fitness curve shows only minimal improvement over generations, with a few small step changes. This suggests that the algorithm struggles to refine solutions effectively. Besides that, the best-found solution of around 7.343 is quite some distance from the best-found solution of 7.326 of a other model, which suggests that the local search mechanisms are ineffective. Exploration: The average fitness persists to be at a distance from the best-found solution. In an ideal scenario, the average fitness would initially have a larger gap from the best fitness, representing diverse solutions. While, over time, the average fitness would converge closer to the best fitness, indicating that the population is clustering around good solutions. This is clearly not the case.

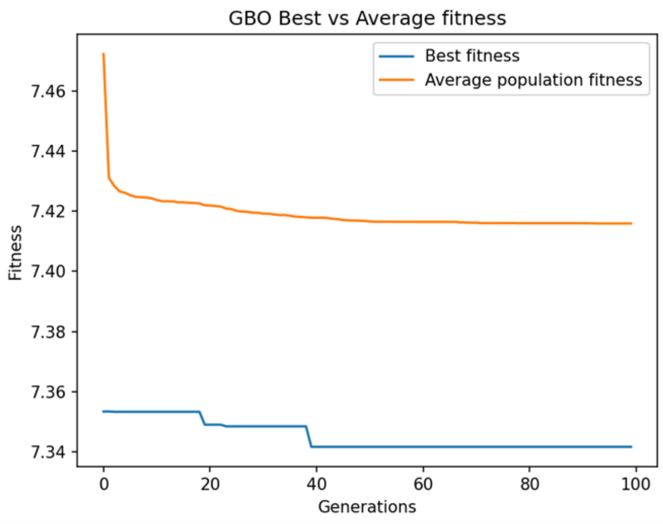

Figure 10.1: Fitness curve GBO Best vs Average

10.2.2 Symbiotic Organisms Search

The *Symbiotic Organisms Search* (SOS) algorithm is another model that has been implemented and modified to fit the C2Pa problem. The meta-heuristic has been developed by (40), which tested the performance of SOS on several unconstrained mathematical problems and structural engineering design problems. Two years later, (41) implemented SOS for the multi-mode resource constrained multi-project scheduling problem, which variant of RCPSP is close to C2Pa. Their algorithm showed excellent performance compared to start-of-the-art models like PSO and GA.

The core concept of the algorithm is symbiosis, a biological phenomenon observed in nature that describes the interaction between two biological organisms. This biological inspiration classifies the algorithm as a bio-inspired algorithm. The implementation of (41) made use of the random key chromosome encoding, which made the implementation straightforward. Given that the algorithm draws inspiration from interactions between organisms in nature, it employs a population-based search strategy. Our implementation used the serial schedule generation scheme, which is different compared to the proposed parallel schedule generation scheme. After decoding, the evolution strategy is executed. This strategy consists of three phases in SOS, namely mutualism, commensalism, and parasitism.

In short, mutualism is an interaction between two organisms which benefits both organisms. Every i-th member of the population is matched with a random member j, this with an beneficial mutualistic relationship in mind. A mutualistic vector which combines the chromosome of both individuals and two benefit factors are constructed. For every member, the orginal chromosome is combined with a random number, the chromosome of the best member of the population, the mutualistic vector, and the benefit factor. The modified organisms are accepted if the fitness is better than the previous. Second, there is commensalism, which is similar to mutualism; however, here, one organism receives benefit where the other is unaffected. Again, the i-th member is matched with a random member j. The candidate solution of i combines the original chromosome, a random number, the chromosome of the best member of the population, and the chromosome of member j. Finally, the parasitism phase is executed. In this phase, the parasite thrives where the host suffers from the parasite or even dies. The i-th member is chosen to act as the parasite vector, which is then modified by mutations. If the mutated member scores better than a randomly selected member j, it replaces member j in the population.

The problems associated with SOS are twofold. The primary issue arises from the computational demands of its evolutionary strategy. For each population member, the algorithm executes a three-phase evolution process. An analysis of these phases reveals that mutualism requires two additional fitness evaluations, while commensalism and parasitism each require one fitness evaluation. This totals four fitness calculations per population member. Considering that fitness evaluation is a computationally expensive operation, these four additional calculations represent a significant inefficiency. The second concern relates to the algorithm's performance. When compared to existing models, SOS fails to demonstrate substantial improvements or advantages. The combination of high computational costs and underwhelming performance suggests that SOS may not be a promising algorithm for further research.

10.3 Flexible time window results SS+LS

Exp.	**Objective**	**Skill match**	**Satisfaction**	**Hourly cost**
	mean ± std dev	**mean ± std dev**	**mean ± std dev**	**mean ± std dev**
TW = 0	-0.2010 ± 0.0053	-1.6272 ± 0.0985	6.099 ± 0.201	151.22 ± 3.25
TW = 1	-0.1993 ± 0.0065	-1.6420 ± 0.1234	6.018 ± 0.241	150.56 ± 3.81
TW = 2	-0.2421 ± 0.0009	-0.9235 ± 0.0760	7.098 ± 0.120	153.96 ± 1.24
TW = 4	-0.2406 ± 0.0030	-0.9704 ± 0.1307	7.083 ± 0.160	152.86 ± 2.64
TW = 8	-0.2542 ± 0.0059	-0.7086 ± 0.1731	7.303 ± 0.151	151.58 ± 4.16

Table 10.1: SS+LS results experiment flexible time windows