Otto-on-Guericke University Magdeburg

Faculty of Computer Science

Master Thesis

Evolutionary Multi-Objective Activity Scheduling

by

Markus Patrick Rothkötter

2022-12-07

Supervision by:

Prof. Dr.-Ing. habil. Sanaz Mostaghim

Chair of Computational Intelligence
Otto-von-Guericke University Magdeburg

Rothkötter, Markus Patrick:

Evolutionary Multi-Objective Activity Scheduling
Master Thesis, Otto-von-Guericke University Magdeburg, 2022

Abstract— Current scheduling systems are built around maximization of efficiency, but rarely take person-specific preferences into account. This work focuses on providing a formalization as well as experimental testing of a scheduling system with individualization aspects. ***Method:*** Using scenario-based use-case considerations, a multi-objective system model is derived. Within this model *a priori* and *a posteriori* preference collection are combined, which creates a guidance direction in search space for the optimization, but does not rule out a trade-off-based decision-making after the optimization. The implementation is based on a combination of NSGA-II and custom operators which aim at a proximity-enhanced mutation and a generation-dependent constraint handling. Experimental evaluation of the system configurations focuses on investigating conflict potential via correlation analysis. Algorithm configurations are evaluated via hypervolumes in objective space and via a basic diversity metric in decision space. ***Results:*** The provided implementation is regarded as a proof-of-principle concerning the intended use case at an abstract level. The measured conflict potential is notably stronger in higher constraint system configurations. Exploration-oriented evolutionary operators perform better in such constraint systems, but cannot provide measurable diversified genotypes, despite generating better or equally good objective values in comparison to a standard implementation. ***Conclusion:*** The proposed method is able to generate diverse schedules, but different algorithm and system configurations may introduce hidden constraints to the problem. These side effects have an influence on the conflict potential of objectives as well as the effectiveness of evolutionary operators which has to be investigated further.

to my family and friends
who always stand by my side and have my back

– · –

Table of Content

List of Figures

List of Tables

1. Introduction

1.1 Motivation for intelligent scheduling

> “ A schedule defends from chaos and whim. A net for catching days. ”
>
> ~Annie Dillard, U.S. author and university lecturer

The term “scheduling” is defined as “the job or activity of planning the times at which particular tasks will be done or events will happen” according to the Oxford Dictionary [1]. Scheduling can be found everywhere: from large-scale infrastructure projects to work coordination within a team up to the individual day planning – in a professional as well as private context.

The process of scheduling multiple, possibly interrelated activities, however, requires time, expertise and experience. Some large-scale tools like PERT require extensive knowledge of the method itself, while agile methods like SCRUM are dependent on the experience of the managing person in charge, as will be introduced in section 2.1.

In a professional context, there may be dedicated experts working on planning, while in a personal context, planning capabilities may be limited by time and a lack of strategies. Enterprise-scale planning does not have a place in most people's daily life. Especially the exploration of different scenarios or scheduling variants in order to find an optimal scheduling can become tedious and time-consuming.

Intelligent methods might have the potential to oversee complex schedules that exceed humans' capabilities to process them effectively. Such methods excel in particular by generating solutions a human planner would not have been able to come up with and when testing different scheduling variants in parallel. An intelligent tool would be of special interest to enterprise users, as illustrated by the finding that 68 % of software projects fail partly or are delayed because of an insufficient reliability of planning [2]. Companies like Autodesk with its tool “Shotgun” [3] already try such methods in order to increase efficiency and provide solutions at enterprise level.

An easy-to-use software would enable private users to leverage advanced methods with little effort and take aspects of the individual's approach towards working into account. Existing software tools like “Reclaim” offer some functionality in this regard by e.g. sorting tasks into time-slots and adding intelligent buffer times around events [4]. However, individual preferences may lead to multiple scheduling options which a user might want to explore and compare in terms of implied trade-offs e.g. between finishing earlier versus having more buffer time. Accounting for individual preferences and weighing trade-offs has not yet received broad attention by current software solutions.

1.2 Motivation for individualised scheduling

The modern world still adheres to the work rhythms of past centuries, which originated from the rise of factory employment and manufacturing. Even though society has transitioned away from manual labour to an office setting with mental labour, approaches towards work schedules such as the eight-hour workday first postulated by Robert Owen in the 1810s [5] still persist today.

The type of exhaustion caused by mental labour is different from the physical stress that is experienced during manual labour, which is why traditional concepts originating from the phase of industrialization are not necessarily suitable for other areas.

There is a plethora of attributes of work that could be addressed and discussed. As a motivation and background for this thesis topic, the following exemplary aspects of mental labour shall be illustrated briefly: cognitive requirements of tasks and varying individual performance influenced by physiological factors.

Attention span – a cognitive factor – is frequently used for guidance during the generation of schedules. Common perceptions concerning attention span assume a predefined threshold for exhaustion of the human brain. This notion, however, is increasingly being questioned in research, which rather suggests a varying, task-dependent attention span [6]. Consequently, assumptions about performance on one type of task may not be applicable to other types, especially when comparing manual and mental labour. In order to support consistent productivity, human-centered planning tools should consider this effect.

Besides individual cognitive factors, there are also physiological factors impacting human performance, such as the circadian rhythm [7] and menstrual cycle [8]. These examples illustrate that there cannot be a uniform blueprint for schedules which is applicable to all individuals while being beneficial for human health at the same time.

Modern working environments try to react to these circumstances. Especially agile methods have the potential to aid by removing rigid schedules and incorporating individualized aspects. Activities can be arranged according to team members' preferences if managing persons, e.g. SCRUM masters, know their team well.

As introduced in the motivation for intelligent scheduling, in a professional context there are dedicated workers who provide experience for balancing all these requirements, whereas there is no direct equivalent in a private context. However, users often know themselves and their working habits, so a dedicated software could aid them in generating custom schedules from these inputs.

A scheduling software solution therefore not only has to incorporate methods of intelligent computation, but also the ideas of multi-objective optimization so that individualization aspects are accounted for.

1.3 Research questions

Before introducing the actual research questions, the question of scheduling in general is classified according to different types of problems. From this general decision, specific research questions are derived and outlined with respect to the topic of individualized scheduling. The present work explores basic concepts in this field, which implies that many advanced aspects of scheduling like varying types of loads are out of scope for a proof-of-principle.
In general, there are three basic types of problems: optimization, identification and simulation problems. As illustrated by figure 1, classification is done according to input, output and the system model.

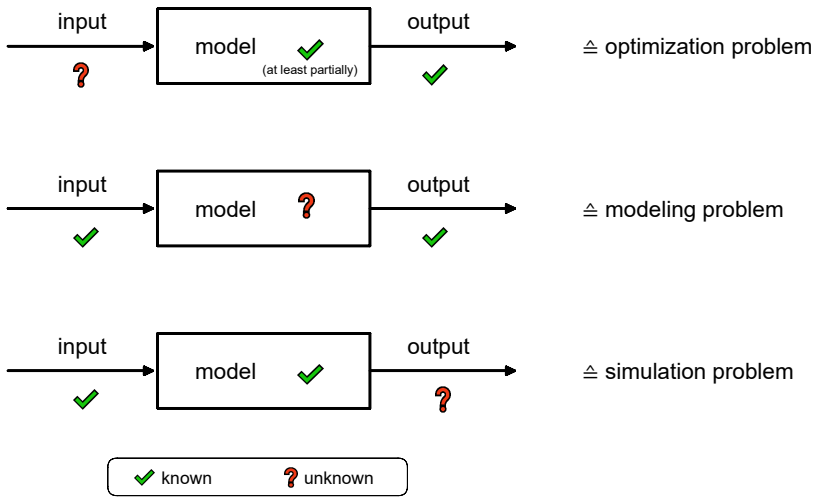

FIGURE 1: Types of problems: optimization, identification and simulation (adapted from [9])

For the use case of planning, the desired output is a schedule that satisfies different criteria, but it is unknown how inputs have to be adjusted to generate the desired output. The given problem is therefore an optimization problem. The goal of the current research is to develop a system model and optimization algorithm which can be used for generating the unknown inputs.
The following research questions are derived to address this goal:

1. How can the given problem of scheduling including individual preferences be formulated as a system model?
2. How can the system model be implement within an optimization algorithm?
3. How can experiments be designed to test the algorithm implementation?
4. How can experimental results be evaluated against user expectations?

Research question one addresses possibilities for modelling the given problem. The overall complexity shall be assessed and a feasible scope shall be defined for the extent of this work. Some typical standard problems of optimization theory are taken into account and shall inspire a custom formal definition which models the given problem.

Research question two is centered around implementing the formalized problem for optimization. An appropriate algorithm has to be selected and checked against use case requirements as well as problem specifics. It is anticipated that some kind of customization has to be made to state-of-the-art approaches in order to make them fully applicable to the current research goals. This implies reviewing different implementation options and taking well-informed design decisions.

Research question three deals with conceptualizing a test bench for experimental evaluation. A suitable set of test data and/or benchmark data has to be selected. Experiments for running tests using this data have to be designed and suitable evaluation metrics have to be chosen.

Research question four focuses on investigating the chosen implementation with respect to functionality and performance. Experiments have to be executed and individual design decisions shall be evaluated. On the basis of these results, the applicability of the system model and optimization algorithm to the use case shall be assessed.

1.4 Structure of presented work

Following the outline of related work, the presented work will describe the development of a proof-of-principle for an intelligent scheduling system with individualization features. The overall structure is closely oriented at the design process of an optimization using meta-heuristics as it was formalized in the book *Metaheuristics*[10].

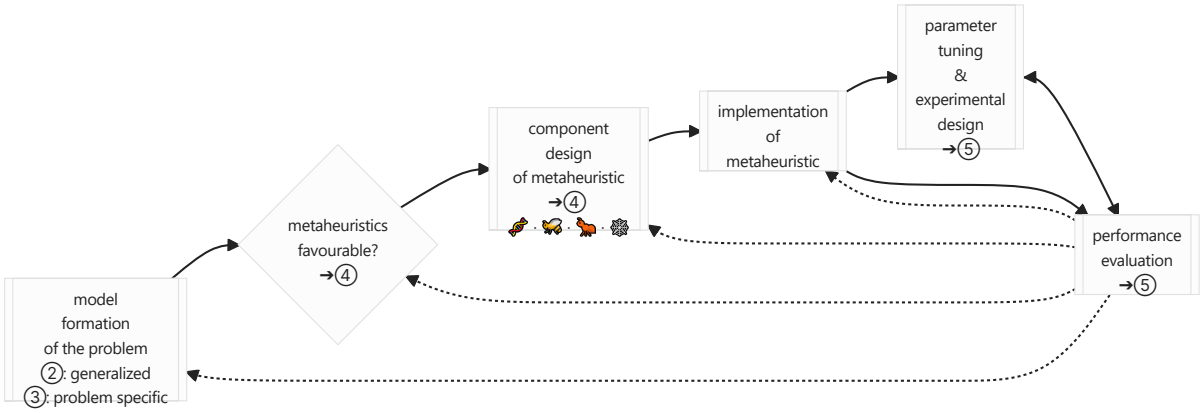

FIGURE 2: Structure of the thesis presented on the basis of a guideline for solving optimization problems (adapted from [10]) successive design flow (–›), iterative feedback-loop (···›), related main chapter (⓪)

The process and thus this thesis structure is illustrated in figure 2 which is a customized interpretation of the process-overview figure provided by TALBI [10]. In order to verify basic functionalities, the main focus lies on the process leading from modelling to implementation. The exact tuning and advanced functionalities are subject to future work.

2. Related work and theoretical background

This chapter gives an overview concerning existing scheduling methods and reflects on approaches towards scheduling from a perspective of computer science. Known standard problems are examined concerning their suitability as a foundation for a custom problem formalization. The bin packing problem (BPP) is used as an example to introduce the topic of optimization. Key elements of multi-objective optimization (MOO) and essential building blocks of evolutionary algorithms are briefly explained to provide relevant background knowledge.

2.1 Overview of scheduling methods

Numerous different scheduling methods have evolved over the past two centuries as a following from the necessity for time planning at industrial sites [11]. To illustrate common features of established scheduling methods, a sub-set of three project management tools is introduced in the following: GANTT, PERT and SCRUM.

Dating back to the early phase of industrialization, the GANTT chart formed one of the first instruments for a structured scheduling approach. This graphical representation introduced an equal division of time and scheduling according to time intervals, which allowed identifying shortages and surpluses of spare time. Additionally, dependencies between units of work created a sequential order for tasks to schedule. [11]

In recent years, the SCRUM framework has gained attention as a scheduling method, especially in the software industry. This method is based on iterative cycles of relatively short working periods called sprints. Within these sprints, tasks are dynamically scheduled by a so-called scrum master who also rearranges them in correspondence to changing circumstances, so that the workload remains manageable for the team. In contrast to other methods, SCRUM describes tasks rather by duration than by static start and end dates. [12]

While the presented methods GANTT and SCRUM inherently provide a single scheduling variant, the PERT method and its further developments aim at taking multiple scenarios into account. This scenario information is collected by specifying a pessimistic, normal and optimistic estimation for the duration of a task. From the sequence information in conjunction with these estimations, different variants of the overall schedule can be derived. [13]

In general, contemporary scheduling methods feature basic common components: Tasks as well as events are described as a time span and are sorted into a sequence either by assigning them into equally-sized, consecutive time slots or by providing dependency information between different tasks.

2.2 Related standard problems and approaches towards intelligent scheduling

This section briefly investigates related standard problems in computer science as well as attempts to solve these via intelligent methods. Analyzing these abstract definitions does not only aid in formulating a new problem, but also allows making first assumptions towards solvability and computational complexity based on known complexity classes of investigated problems.

Thus far, the individualization feature as a key aspect of the presented work does not play a major role in research on intelligent workforce management. The main focus of current research is efficiency and ensuring alignment of conflicting schedules. While this type of planning is mostly based around hard factors such as worker availability, some research has included soft factors such as skill level and workers' preferences concerning team composition [2]. However, the focus here is on groups of workers instead of scheduling for a single person, which is why research in the area of workforce management is less applicable in this context. Therefore, the further search for related work is targeted towards scheduling in general without the aspect of individualization.

In order to find related standard problems, a search is made for analogies of different components of scheduling in standard problems. From an abstract perspective, scheduling consists of assigning items with specific properties (events/tasks/activities) to distinct containers (time slots/sprints). Relationships between different items to schedule indicate a specific sequence, which translates into traversing nodes in a graph in a specific order.

With respect to the scope of developing an intelligent scheduling tool for a single person, specifying complex dependency trees of items is not considered part of the standard use case covered in a proof-of-principle. Consequently, many of the well-studied problems from the area of job scheduling cannot be considered. Standard problems are frequently built around multiple machines and/or sequential tasks like the typical *job shop problem* or *flow shop problem* [14], which does not align well with the goals of single-user schedule generation.

A problem which does present large overlaps with the intended use case is the *knapsack problem*, which can be considered as a generalized form of the *single machine scheduling problem* [14] and is focused on maximization of efficiency by packing the most profitable set of items into a single container, the knapsack. However, scheduling methods like SCRUM are commonly built around multiple containers, i.e. equally divided time units, as detailed in section 2.1. Relaxing the single-container constraint leads to the generalized problem class of *bin packing problems* (BPP), which is considered applicable as the foundation for the intended use case. The decision for BPP as a foundation is backed by research that shows similar applications in e.g. multi-objective scheduling of micro-services within a network [15].

In the following, BPP is used for illustrating the optimization in general and serves as a starting point for formalizing a custom problem reflecting the use case of individualized scheduling.

2.3 Bin Packing Problem for illustrating optimization

The BBP is a combinatorial, NP-hard [16] problem in computer-science and has been very well-studied in different contexts. In this work, the base definition of a one-dimensional BBP formulated by MARCELLO is used. All equations and formalizations in this section are taken from [17].

The first step for dealing with an optimization problem is to define a system model and encoding. The basic one-dimensional BBP considers a group of items (see equation 1) which have to be sorted into containers (bins, see equation 2) with a fixed loading capacity. Each item has a specific weight w_j, the total weight of all items within one bin must not exceed the capacity of a bin *c*. The target of an optimisation is called objective and is quantified by a so-called objective function [10]. In case of the BPP, this is a minimization of the number of bins used which can be formulated like in equation 3.

$$x_{ij} = \begin{cases} 1 & \text{item } j \text{ is in bin } i \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \ldots, u \quad j = 1, \ldots, n \tag{1}$$

$$y_i = \begin{cases} 1 & \text{solution uses bin } i \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, \ldots, u \tag{2}$$

$$f: \quad \min \sum_{i=1}^{u} y_i \tag{3}$$

Optimization procedures might generate solutions which have a good value of the objective function, but are not useful for solving the actual optimization target as the solution is in conflict with the restrictions of the underlying system, so-called constraints. The constraint functions describe the bounds of the solution space which limit the feasible regions. From a mathematical perspective, infeasibility can be defined as the violation of defined bounds by the values of one or more constraint functions [18]. Concerning the optimization process, [19] provides an intuitive explanation: Constraints can be considered as hard objectives which have to be satisfied before optimizing the actual objectives.

Commonly, two constraint types are distinguished: Inequality (g) and equality (h) constraints. While inequality constraints allow a region of valid values like the maximum weight constraint of the BPP does (see equation 4), equality constraints have an exact value specified that has to be met [10]. For the BPP-example, the *0-1*-condition is a typical equality constraint [17]: An item cannot be divided and must not occur in more than one bin at a time. Formally, this constraint breaks down to equation 5.

$$g: \quad \sum_{j=1}^{n} w_j x_{ij} \leq c y_i \qquad (i = 1, \ldots, u) \tag{4}$$

$$h: \quad \sum_{i=1}^{u} x_{ij} = 1 \qquad (j = 1, \ldots, n) \tag{5}$$

In the case where a solution is infeasible, [20] specifies two general types of quantifying the degree of infeasibility:

- *sum of infeasibilities* (SINF)
  ≙ cumulative amount of constraint violations above all bounds
- *number of infeasibilities* (NINF)
  ≙ cumulative count of occurrence of constraint violation

Quantifying the degree of infeasibility is an important input for constraint handling, which is the process of dealing with solutions that violate constraints. Several approaches are known in the literature, according to [19], the best known examples are:

1. aiming at rejection of solutions by selection mechanisms
2. mapping the decision space to feasible regions only
3. tolerating infeasible solutions, but penalizing them
4. rescaling the objective function in order to rank feasible regions higher

Deciding on a suitable constraint-handling mechanism is essential for successful optimization, especially when dealing with real-world problems.

2.4 Multi-objective optimization problems

The previously described, original BPP is a so-called *single-objective problem*. Real-world problems, however, are rarely reducible to a single objective. Consequently, the class of *multi-objective problems* exists to address this issue. Key element of an MOP are conflicting objectives, i.e. objectives that are structured in such a way, that improving the value of one objective will worsen the value of another objective [21]. The general structure of the definition at system level remains the same, except for replacing a single objective function with multiple ones.

Despite the fact that different values have to be evaluated, fundamental statements about the quality of solutions can be made by leveraging the *pareto dominance*, as defined in equation 6. The set of non-dominated solutions is called *pareto front*. [21]

$$\underbrace{x \prec y}_{x \text{ domintes } y} \quad \Leftrightarrow$$

$$\underbrace{f_i(x) \leq f_i(y) \;\; ,\forall i \in \{1,\dots,m\}}_{\text{I: } x \text{ is equal to or better than } y \text{ for ALL objectives}} \tag{6}$$

$$\wedge$$

$$\underbrace{\exists j : f_j(x) < f_j(y)}_{\text{II: } x \text{ is better than } y \text{ for at least ONE objective}}$$

Since MOPs consist of more than one objective function, a performance comparison of different solution sets via absolute objective values does not do justice to the actual system nature and does not allow establishing a ranking of the solutions. Consequently, a performance indicator is needed that takes the real distribution in objective space into account. The *hypervolume* metric (HV) and *inverted generational distance* (IGD) are two recent measures that not only focus on the objective values, but also on the diversity of the pareto front [22]. Detailed descriptions concerning the application of metrics in this work are given in the evaluation section 5.2.

Solving MOPs is mainly centred around two disciplines, according to [23]: *multi-criteria decision making* (MCDM) and *multi-objective optimization* (MOO). While MCDM focuses on retrieving one solution acceptable to the specified preferences of a decision maker, MOO targets an approximation of the pareto front and lets the users decided afterwards, according to their preferences. These two modes of preference collection also label the general approaches towards user interaction: *a priori* and *a posteriori*. Combinations of these two modes are also subject to current research. Due to the continuous user interaction, they are labeled *interactive* or *progressive* approaches. [21]

2.5 Methods towards solving optimization problems

While a multitude of optimization methods exists, as shown in figure 3, the presented work focuses on *meta-heuristics*. This class of optimization procedures provides acceptable solutions for hard and complex problems in a reasonable time without guaranteeing optimality [10]. A *meta-heuristic* can be understood as a general design pattern or template that can be applied to almost any optimization problem [24], even with problems for which neither an exact methods nor specific heuristics exist.

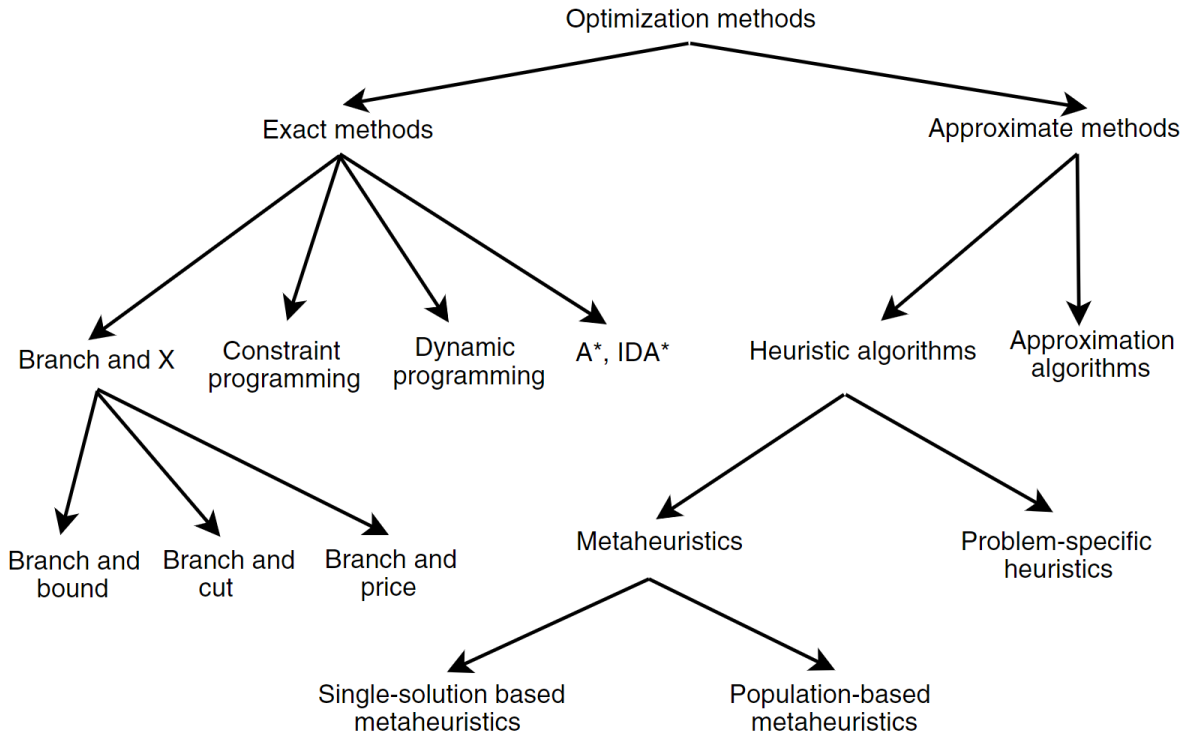

FIGURE 3: Overview of classical optimization methods [10]

Despite being applicable to almost any optimization problem, *meta-heuristics* are no full replacement for exact or problem-specific methods. Small and even large, but well-structured problem instances might be solved better using a specific heuristic or an exact approach [10], despite their computational complexity. Consequently, it has to be assessed carefully whether applying *meta-heuristics* is reasonable for a given problem.

For the scheduling problem under investigation, it is assumed that this problem is NP-hard, since the BPP is proven to be NP-hard [16] and there are strong similarities between both problems. Even though multiple specific heuristics for the BPP have been developed, these are not applicable for the given problem, as most scheduling systems contain more features than the original BPP model of bins and items. Additionally, the problem instances may differ strongly in their size and the overall goal of an individualized scheduling is to provide multiple variants to chose from. This requirement cannot be satisfied with ease in specific heuristics, but population-based *meta-heuristics* open wide-ranging possibilities for MOO. So, the application of MOO is reasonable.

2.6 Building blocks of evolutionary algorithms

Evolutionary algorithms are a class of population-based meta-heuristics [10], among the best-known of which are the genetic algorithms developed by HOLLAND in 1975 [25]. Even advanced algorithms of this class follow the same fundamental design patterns described briefly in this section.

The fundamental idea of genetic algorithms is to sample a random population of individuals where each individual forms a solution to the problem [26]. The specific values of decision variables for one solution of the problem are encoded in the so-called genotype of an individual, while the corresponding appearance in solution space is considered as phenotype. The associated performance value of the phenotype represents the fitness of an individual (fitness function), therefore the objective space of the problem can be seen as a fitness landscape. Based on this fitness function, the algorithm mimics the DARWINIAN theory of evolution. [25]

Inspired by the concept of *survival of the fittest*, individuals are selected for recombination operations to produce new generations of individuals. These so-called cross-over operations serve as a mechanism for exploitation of the solution space, aiming to obtain better solutions by recombining fit individuals in order to produce an even fitter offspring generation [27]. Random manipulation of the genotype of some individuals by mutation operators acts as a counteractive exploration mechanism [27]. Individuals are run through this overall procedure until a defined termination criterion is reached, like shown in figure 4. Within the process, the application of genetic operators might create individuals that violate the system constraints. For such cases, an optional operator for repairing those individuals can be established.

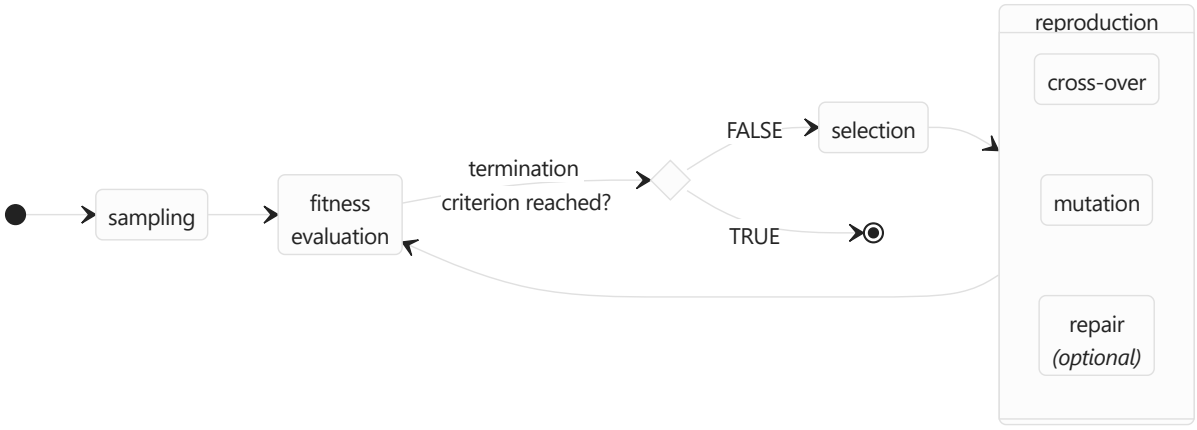

FIGURE 4: Top-level process of a genetic algorithm

3. Problem description

3.1 Differentiation from existing approaches

In contrast to the examples for scheduling methods shown in section 2.1, the present work focuses on developing an approach that enables individual users to generate sophisticated schedules without having to leverage enterprise-level methodologies. Differing from existing attempts for intelligent scheduling 2.2, individualized aspects of scheduling form a key component of the present work. While other methods focus on optimizing schedules based on fixed technical measures, optimization targets in this work shall reflect the diversity of working habits.

While a common focus of scheduling solutions is productivity, the present work aims to incorporate an individualized, human-centered approach. Instead of targeting only maximum efficiency, workers shall benefit from plans which respect their unique needs in terms of health and well-being. An important goal is therefore to go beyond a single prescribed optimization outcome: A spectrum of possible solutions enables users to choose according to situational circumstances and trade-off considerations. Thus, multi-objective optimization is considered a suitable approach.

3.2 Proposed method: Activity-based scheduling

This section will introduce a planned scheduling method, giving a brief overview of its components and their relations. Aspects of individual component related to individualized scheduling will be detailed in the formalization (3.3+) and implementation (4) sections.

Current scheduling solutions sometimes fail as a real-world workflow in everyday life because the user has to think about both content to schedule and the scheduling sequence at the same time. The proposed method aims to lift this burden by requiring two basic inputs only: the user's availability (time slots) and things to be done (activities). The core concept therefore separates concerns and leaves the complex challenge of arranging both – the actual scheduling process – to the intelligent algorithm.

Time slots are supposed to feature continuous work and are therefore called *sprints* (also SPR). This term also occurs in the traditional SCRUM framework [12], but often refers to timespans in the range of weeks. For the current abstract model, no specific time unit is considered. Nevertheless, the overall extent of the schedule shall be defined as *planning horizon* and is given as the fixed total number of sprints (n_{SPR}).

During the scheduling, activities (ACT) are assigned dynamically to sprints up to a defined maximum capacity ($c_{\mathrm{SPR;\,max}}$) of the sprint. The level of utilization shall be called *fill grade* ($v_{\mathrm{fill\text{-}grade}}$) and is equivalent to the total sum of durations of all activities within this sprint.

Activities themselves are not defined by start and end points in time, in contrast to some conventional project scheduling methods (section 2.1), but are solely defined by the duration of

activities and an optional external deadline, so that the user does not input any bias towards timing activities.

In a real-world use case the duration of an activity has to be estimated by the users themselves. Humans tend to have a good estimation of smaller-sized magnitudes with a rapidly increasing error rate when magnitudes are expanding [28]. Inspired by this fact, activities shall only be sized between a fraction of sprint length or total sprint length.

Within the schedule, there must not be activities without a planned duration, inspired by the observation that large project schedules often fail because of small tasks. These minor tasks are assumed to be too small for scheduling, but accumulate in the end, leading to major delays in the overall schedule [28]. Additionally, the proposed method assumes that a user can only execute one activity at a time, as parallel scheduling would prevent the clear aggregation of activity durations.

The previously described components are the basics needed to form an abstract proof-of-principle model, but do not cover advanced features required for a real-world product. The initial challenge of arranging activities into time slots is already assumed to be in the class of NP-hard problems, as detailed before (2.3). For the sake of a feasible scope of the current work, advanced features of (individualized) scheduling are only mentioned here shortly and omitted afterwards intentionally.

Full-scale scheduling methods cover relations between different, dependent activities and create conditions based on the status of each activity, e.g. as with PERT. Additionally, the introduction of a classification for activities could account for different types of loads and provide a human-centred approach of scheduling these. Incorporating team aspects would further enrich the possibilities of a scheduling solution, especially concerning fairness in decision-making in case of conflicting individual preferences. Such extensions of the system are left for future work.

3.3 Formal system description towards optimisation

The following sections focus on understanding the basic system model and the process of model generation. To this end, the methodology for deriving the system model and a novel approach to preference collection are presented. The real-world system is evaluated in terms of component structure and a classification of requirements concerning infeasibility is introduced.

3.3.1 Basic methodology for deriving the formalization

In contrast to other known problems like the BPP, which was described in section 2.3, the current problem does not have a pre-described formalization. Some real-world problems have objectives which are obvious at first glance, so-called *self-sufficient objective functions* [10]. For example, in route-planning the minimization of the overall distance is very likely to be an objective.

The given problem has many different subjective, user-dependent aspects as each human has a different approach towards working as explained in section 1.1. Resulting from this,

it is clear that there are attributes of the problem where it is unclear what the user wants to optimise for. For example, the project buffer could be increased to incorporate last-minute edits or decreased to have a tighter schedule. The user's opinion might even change because of situational circumstances or different combinations of attributes. These uncertainties are complicating factors in the formalization process.
Respecting the user-specific aspects within the planned optimization implies using a formalization process which takes the user's questions into account. In this work, the general methodology of "User-Centric Design" is applied, which is standardized in ISO 13407 "*Human-centred design processes for interactive systems*" [29] and can be summarized in three key phases:

1. Identify the potential users
2. Specify the intended use case
3. Describe the circumstances of usage

The outcome of running through these phases are user stories for individual users, which are subsequently generalized to different scenarios. From these scenarios, relevant system attributes (e.g. remaining time to deadline) shall be identified and desired attribute values for different user groups shall be envisioned. For example, one user group might prefer finishing well ahead of the deadline while others gain a productivity boost from time pressure.
This type of user analysis serves as the basis for the following formalization, but it is not described in detail due to the scope of this work. However, each resulting generalized scenario is laid out within section 3.3+ to illustrate the origin of the formalized elements, i.e. objective and constraint functions as well as preferences.
As a first step in the formalisation process, a basic system model is derived from the scenarios as shown in figure 5 on the left. The system model consists of decision variables and an encoding function [23].

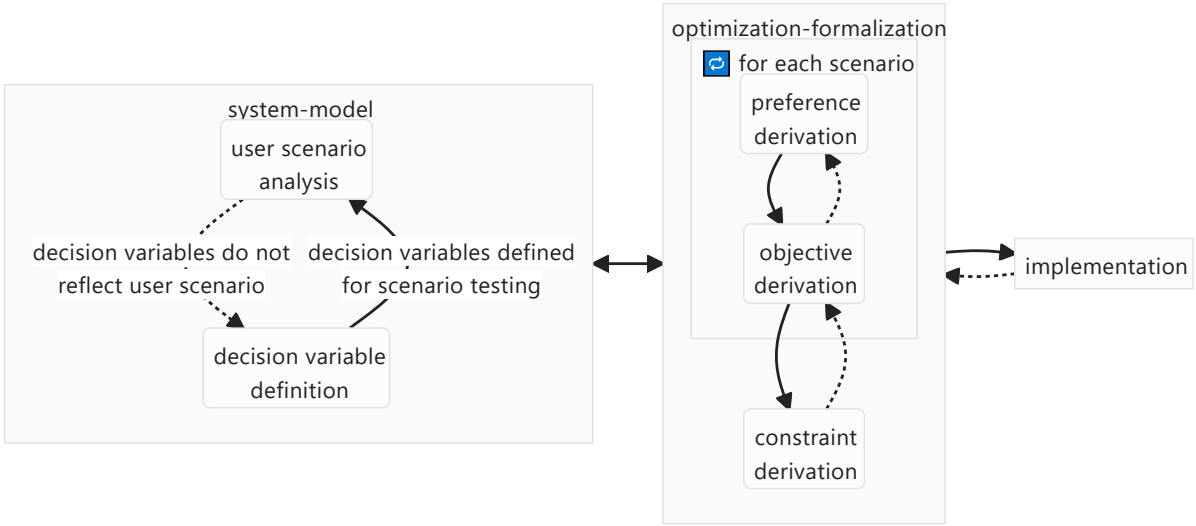

FIGURE 5: Top-level process description of the modelling process

As soon as a model which depicts all elements of the usage scenarios is found, the actual optimization can be formalised. The starting point are preferences derived from system attribute values desired by the user. As shown in the center of figure 5, preferences form the basis for formulating objectives which span all potential expressions of desired preference values. Constraints are derived based on a consideration of system limitations envisioned in the use-case scenarios as well as potential trends of objective function values. This approach is developed as it is expected that the real-world system structures under consideration are not self-evident.

Even though the previously described process may give the impression of a fully top-down design process, it is important to note that this is not the case. The entire procedure is supposed to be iterative, as approaches to designing an optimization frequently are [10]. Potential designs often contain assumptions which have to stand up to experimental testing. In the presented work, special attention will be paid to the following attributes of components:

- **conflict potential of objectives** ➔ The fundamental aspect of MOP are conflicting objectives which are necessary for generating a diverse solution space.
- **strictness of constraints** ➔ If constraints are defined too strictly, a problem might become unsolvable. In order to ensure solvability, constraints might have to be relaxed and violations have to be handled.
- **approach to preference collection** ➔ Different ways of collecting preferences might require different problem representations so that they can exert their full power. Changing the collection approach might require a full reformulation of the problem.

3.3.2 Specific considerations regarding preference collection

As described in section 2.3.1, there are two main disciplines for solving MOPs. In the current use case, the algorithm shall present the user with multiple different schedule variants to choose from. Users' diverging general attitudes result in fundamentally different values of system attributes, e.g. one user might want to maximize spare time as a buffer for potential delays, while another user wants to minimize spare time for a more efficient schedule.

These strong opposites lead to a very broad variant space of solutions. The broad space might complicate the optimization, as strong explorative behavior is required, which in turn leads to less exploitative behavior. This impression is backed by the results of preliminary test runs, which did not generate sufficient results when trying to incorporate the whole spectrum of potential values of system attributes. So, an approach towards optimization is needed that limits the prevalent size of the solution space, while still providing a diverse set of solution variants.

In first experiments, it was attempted to reflect all possible attitudes towards work in one optimization. However, from a use-case perspective this all-encompassing attempt might not be necessary. Each user has an opinion concerning required work circumstances, which is rather stable as it reflects a general habit or mood. Consequently, an optimization approach that guides the algorithm into this general direction may present an opportunity for generating more suitable results.

This line of thought has some similarities with the established approach of using *reference points* in interactive MOO, which aims at retrieving "… a more manageable subset of solutions (decisions) that correspond best to user's preferences" [21, p.39]. In this established approach, not only user preferences are considered as configuration input for the functions of the system, but also external impact factors from the environment [21]. Yet, implementing such a highly granular approach seemed too advanced in the early stages of the system design of the current work, so a custom less complex version was developed.

In this custom approach, aspects of MCDM and MOO are combined by manipulating the objective functions by a user-defined preference. The preference collected *a priori* shall be labeled *preset preference*, in order not to be confused with a reference point in objective space. Details concerning the resulting objective formalization are given in section 4. It is hoped that this way of incorporating preferences allows local exploitation in a limited solution space in contrast to the standard *a priori* as well as *a posteriori* approaches, as shown in figure 6.

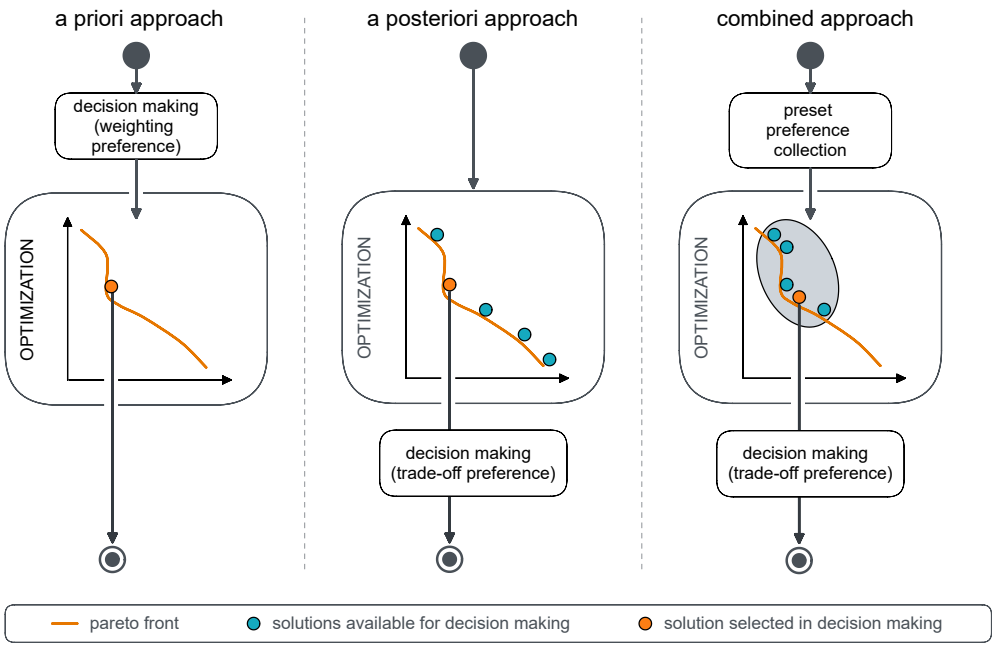

FIGURE 6: Schematic overview of preference collection approaches

3.3.3 Specific design considerations regarding infeasibility

Meta-heuristics are likely to generate solutions which are infeasible (see section 2.3) due to the population-based and randomness-based approach. Poor solutions might steer the population away from feasible regions and the evaluation of infeasible solutions can be computationally costly, impacting the overall efficiency. These effects imply that design considerations concerning infeasibility and constraint handling are crucial for a successful optimization.

Taking well-informed decisions towards the implementation of constraint handling requires a solid understanding of the real-world effect of constraints as well as the implications of constraint-handling methods concerning the optimization process. Some problems have optimal solutions at the edge of infeasibility [30], which means keeping solutions violating constraints can aid in finding these.

The common definition of feasibility as introduced in 2.3 logically results in a binary classification: feasible vs infeasible solutions. Real-world problems, however, often require a higher degree of granularity to distinguish whether a solution is useful or not.

On the one hand, some solutions are impossible in the context of the real-world system e.g. due to physical limitations. On the other hand, some solutions simply have unfavorable values and thus do not meet the performance target. For mathematical and benchmark problems, ruling out infeasible solutions might be a valid approach, but in real-world applications a suboptimal solution might be better than no solution at all. This

is especially true for circumstances in which e.g. the available computation time is limited.
In order to reflect these nuances of feasibility, this work proposes an extended classification:

- **feasible solution**
  ≙ solutions within constraint bounds and therefore in the desired regions
- **infeasible solution**
  ≙ solutions violating constraint values, but still usable with respect to real-world circumstances
- **inviable solution**
  ≙ solutions violating premises of the real-world system thus, unusable in general

Meta-heuristics are typically considered as “any-time” algorithms [9], a description which shall be assessed further with respect to the newly proposed classification of feasibility. The term “any-time algorithm” describes procedures which at each iteration supply a solution that is superior to solutions from the previous iterations – the algorithm can be stopped at any time and will provide an improved result [31].
The original any-time algorithm definition is not very specific concerning potential constraint violations. A solution with an improved objective value might still violate constraints. Real-world constraints regularly reflect physical limitations or are part of error-prevention measures which shall avert critical failure. Solutions violating these constraints have to be considered as inviable solutions according to the introduced classification.
Some applications do not provide enough time to wait for evolutionary pressure to cause the extinction of inviable solutions, e.g. in real-time applications. Additionally, new inviable solutions might reoccur during application of the algorithm’s operators. In case the algorithm must be stoppable and must provide a viable solution at any time, this can result in a new design requirement: *viable at any time*. This requirement translates to two components: tolerate infeasible solutions, but block inviable solutions at any time.
Establishing these conventions might seem abstract at first glance. Even though meta-heuristics are applicable to almost any optimization problem [10], operators frequently have to be adapted to the specific real-world problem. Having a clear classification of constraints during the formalization streamlines the algorithm design and lays the foundation for justifying design choices.

3.3.4 Observation models for a multi level problem

The given problem, as described in section 3.2, exposes a nested structure of system components as illustrated in figure 7: activities are contained within sprints which form a schedule.

FIGURE 7: Levels of containment of given scheduling problem

Optimization in technical systems often has multiple design levels [32]. The multi-level structure of the system implies that interactions on one level might trigger cascading changes onto contained ones. An observation level shall be introduced here, inspired by the common practice of defining an observation model in sorting algorithms [9], which indicates the ordering element of a system. By specifying the observation level ahead of making design decisions, an attempt is made to predict potential side effects during the formalization. Nevertheless, it might not be possible to foresee all interrelations in their entirety at an early design stage, which is why an iterative design is required (as mentioned in section 3.3.1).

3.3.5 System model, decision variables and optimization problem

The selection of the decision variable is highly problem-specific. According to the defined scope, the proposed scheduling method is closely related to the established scheduling method SCRUM as described in 2.1. Basically, each activity is assigned to a sprint (time slot) – the order of activities inside a given sprint does not matter. The inputs to the optimization problem are lists of sprints, activities and durations as well as deadlines of activities.

An instance of the scheduling problem can therefore be described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{D}, m_{\mathcal{D}}, m_{\text{deadline}})$. The system model is built around an index set of sprints $\mathcal{S}$, containing the sequential integers, so that $\mathcal{S} = \{0, \ldots, n_{\text{SPR}} - 1\}$. As the sprints are in a chronological order in time, operations and relations of $\mathbb{Z}^{0+}$ apply. Activities to schedule are defined in an index set $\mathcal{A}$, containing the sequential integers $\mathcal{A} = \{0, \ldots, n_{\text{ACT}} - 1\}$. For each activity, two attributes are encoded in mapping functions: Firstly, the surjective

mapping function $m_{\text{deadline}} : \mathcal{A} \mapsto \mathcal{S}$ associates a sprint $x_{\text{SPR}} \in \mathcal{S}$ as deadline for each activity $x_{\text{ACT}} \in \mathcal{A}$. Secondly, the surjective function $m_{\mathcal{D}} : \mathcal{A} \mapsto \mathcal{D}$ provides an activity duration obtained from the set of duration information $\mathcal{D}$.
The actual search problem is finding a mapping for assigning activities to sprints, which is defined as a mapping function in equation 7. This function is total, since the *activity to sprint* cardinality is 1, but it is neither surjective nor injective, since the *sprint to activity* cardinality is 0..*.

$$m : \underset{|\mathcal{A}|=n_{\text{ACT}}}{\mathcal{A}} \mapsto \underset{|\mathcal{S}|=n_{\text{SPR}}}{\mathcal{S}} \qquad \text{with} \qquad \begin{array}{rcl} \forall x_{\text{ACT,i}} & : & x_{\text{ACT,i}} \in \mathcal{A} \\ \forall x_{\text{SPR,i}} & : & x_{\text{SPR,i}} \in \mathcal{S} \end{array} \tag{7}$$

This mapping forms the decision variables and therefore the input to the optimization problem, which is defined in equation 8.

$$\begin{aligned} \underset{m}{\text{minimize}} \quad & (f_{\text{MSL}}(m), f_{\text{MSC}}(m), f_{\text{DLP}}(m), f_{\text{DDB}}(m), f_{\text{DTB}}(m)) \\ \text{s.t.} \quad & 0 \geq \; g_{\text{MAF; UAA}}(m) \\ & 0 \geq \; g_{\text{MAF; USC}}(m) \\ & 0 = \; h_{\text{NDF}}(m) \\ & 1 = \; h_{\text{UQA}}(m, x_{\text{ACT}}) \qquad \forall x_{\text{ACT}} \in \mathcal{A} \end{aligned} \tag{8}$$

As shown above, the overall definition contains five objectives and four constraints, which will be introduced in the following sections.

3.4 Formalisation of base scenario

3.4.1 Base scenario description and derived preferences

The base scenario features the simplest form of a project plan: Within a fixed planning horizon, a fixed number of activities has to be assigned to available time slots. Depending on the intended use case, this basic system model leads to two closely related preferences. The first preference is imagined in a broad context and contains the wish to finish at the earliest date possible, resulting in the shortest sequence of sprints. The second preference originates from the specific context of billed time slots. A user wants to achieve the smallest number of sprints to pay, which implies that as few sprints as possible are used.
These preference descriptions might already indicate some overlaps, which are going to be investigated in the following objective formalization.

3.4.2 Derived objective: Minimum Schedule Length (MSL)

Using the schedule as the observation level indicates a self-sufficient objective: Minimum Schedule Length, abbreviated MSL in the following. This objective originates from the user's preference that the last occupied sprint shall occur as early as possible. The fill-grade of a sprint $v_{\text{fill-grade}}$ as defined in equation 9 determines how much of the capacity of a sprint is occupied.

$$v_{\text{fill-grade}}(x_{\text{SPR}}, m) = \sum_{\substack{x_{\text{ACT; i}} \in \\ \{x_{\text{ACT}} | m(x_{\text{ACT}}) = x_{\text{SPR}}\}}} m_{\mathcal{D}}(x_{\text{ACT; i}}) \tag{9}$$

A system premise defined here is that a schedule always begins with the first sprint, no matter if it is occupied or not. As the schedule is defined as an ordered sequence of sprints, the schedule duration f_{MSL} is equivalent to the index value of the last non-empty sprint. The resulting objective is defined formally in equation 10.

$$f_{\text{MSL}}(m) = \left[\max \left\{ x_{\text{SPR}} \in \mathcal{S} \mid v_{\text{fill-grade}}(x_{\text{SPR}}, m) \neq 0 \right\} \right] \tag{10}$$

3.4.3 Derived objective: Minimum Sprint Count (MSC)

Considering the observation model of sprints, it does not matter where these sprints are located in the schedule. Therefore, the shortest schedule in this case is equivalent to the

desired outcome of the second objective: minimizing the count of occupied sprints, abbreviated as MSC.
In a simple setting without further objectives, introducing MSC is partially redundant to the previously introduced MSL objective. The solutions towards MSL are a sub-set of the ones applicable to MSC. However, preliminary testing showed interrelations with other objectives, so these two definitions at different observation levels were created, aiming at providing a future-proof approach at the price of some overlaps in the initial design phase. Formally, the *MSC* objective is defined in equation 11, which can be described as counting the number of non-empty sprints.

$$f_{\text{MSC}}(m) = |\{x_{\text{SPR}} \in \mathcal{S} \mid v_{\text{fill-grade}}(x_{\text{SPR}}, m) \neq 0\}| \tag{11}$$

3.4.4 Derived constraint：Unique Assignment (UQA)

The most fundamental constraint assumes atomicity of activities, which means an activity can neither be split nor partially scheduled. In addition, the system model assumes that each activity has to be scheduled, but must not be placed in more than one sprint. This constraint shall be labeled “Unique Assignment”, UQA for short.
The same constraint exists for the BPP introduced in the form of a “0-1” constraint originating from linear programming (see equation 5). This equation, which describes the binary mapping function between a bin and each container, shall be true exactly once. In the given problem this aspect shall be formulated as an equality constraint without possibility of relaxation (see equation 12).

$$h_{\text{UQA}}(m, x_{\text{ACT}}): \quad 1 = |\{x_{\text{SPR}} \in \mathcal{S} \mid m(x_{\text{ACT}}) = x_{\text{SPR}}\}| \qquad \forall x_{\text{ACT}} \in \mathcal{A} \tag{12}$$

3.4.5 Derived constraint：Maximum Sprint Fill grade (MAF)

Similar to the BPP , there is a limited capacity for the container element, in this case the sprint. The resulting restriction shall be labeled “Maximum Sprint Fill grade” (MAF).
In the context of the definition of this system, a MAF-constraint violation is considered to make the whole solution inviable, justified by the use-case: Scheduling too an insufficient number of activities within a sprint merely leads to idle time, but scheduling too many activities leads to a lack of time, which in turn requires a selection decision between activities in order to get back on track. There is thus no clearance area which allows adjustments that results in requiring a strict constraint enforcement.

The constraint is consequently defined as an equality constraint given in equation 14 and defined with the help of the indicator function e (equation 13.

$$e_{\text{overload}}(x_{\text{SPR}}, m) = \begin{cases} 1 & , c_{\text{SPR; max}} < v_{\text{fill-grade}}(x_{\text{SPR}}, m) \\ 0 & , \text{otherwise} \end{cases} \tag{13}$$

$$h_{\text{MAF}}(m): \quad 0 = \sum_{x_{\text{SPR; i}} \in \mathcal{S}} e_{\text{overload}}(x_{\text{SPR,i}}, m) \tag{14}$$

3.4.6 Derived constraint: Minimum Fill grade Constraint (MIF)

The traditional BBP as introduced in section 2.3 only specifies restrictions on the container's maximum fill-grade. The given problem however, contains real-world constraints that require a specified minimum fill-grade: For example, it is not reasonable for commuters to commute because of a time slot with only some allocated activities.
Introducing such a constraint heavily restricts the space of feasible solutions. Solutions which might have been valid according to BPP specifications are not valid now, as illustrated by figure 8.

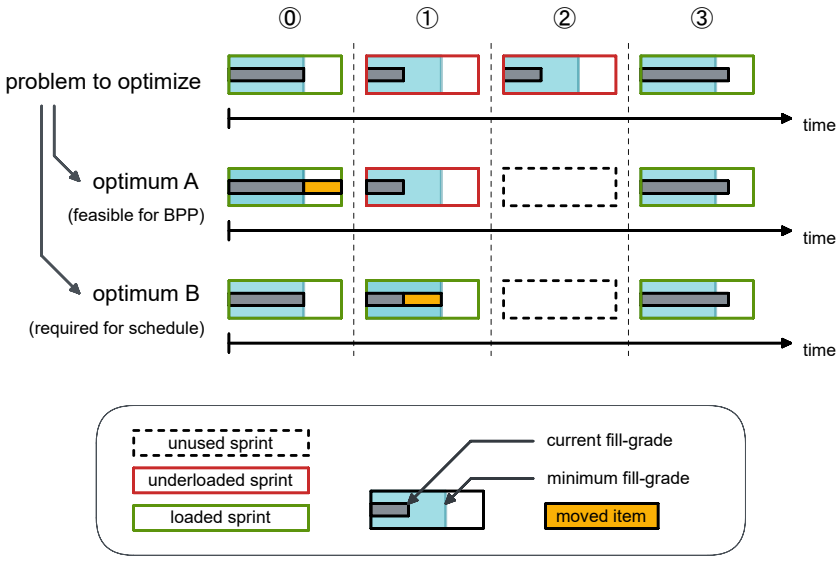

FIGURE 8: Minimum fill-grade constraint restricting solution space compared to BBP, using a bullet chart for visualization

In the following, a sprint which does not satisfy the minimum fill grade constraint shall be called an “underloaded” sprint in correspondence with the “overloaded” sprint as

defined in 3.4.5. Scheduling too few activities generates idle time, but will not crash a schedule. In contrast to the MAF constraint introduced before, constraint violations are considered to produce infeasible, but viable solutions. Consequently, a non-rejecting mechanism for constraint handling and a measure for constraint violation have to be established.
As explained in the introduction on constraints (see section 2.3), different ways of measuring constraint violations exist. For the current constraint, this implies there are two different approaches towards formalization:

1. number of underloaded sprints
2. amount of underloading ≙ total duration of activities residing in underloaded sprints

Both variants detect different types of infeasible phenotypes. The solution space is limited quite heavily and it is very likely that these infeasible phenotypes occur especially during sampling. This assumption has been confirmed in preliminary tests, so both variants are used in the following (equation 16, 17). The strictness of constraints is relaxed by choosing a formalization as inequality constraint. Restricting bounds are described by threshold values ($v_{\text{th; USC}}$ ≙ *underloaded sprint count*, $v_{\text{th; UAA}}$ ≙ *underloaded activity amount*) which will be detailed in section 4.6 .

$$e_{\text{underload}}(x_{\text{SPR}}, m) = \begin{cases} 1 & , c_{\text{SPR; min}} > v_{\text{fill-grade}}(x_{\text{SPR}}, m) \\ 0 & , \text{otherwise} \end{cases} \tag{15}$$

$$g_{\text{MIF; USC}}(m): \quad 0 \geq \sum_{x_{\text{SPR; i}} \in \mathcal{S}} e_{\text{underload}}(x_{\text{SPR; i}}, m) - v_{\text{th; USC}} \tag{16}$$

$$g_{\text{MIF; UAA}}(m): \quad 0 \geq \sum_{\substack{x_{\text{SPR; i}} \in \\ \{x_{\text{SPR}} \in \mathcal{S} | \\ e_{\text{underload}}(x_{\text{SPR}}, m) \\ = 1\}}} \left(c_{\text{SPR; min}} - v_{\text{fill-grade}}(x_{\text{SPR; i}}, m) \right) - v_{\text{th; UAA}} \tag{17}$$

3.5 Formalisation of Load Distribution Scenario

3.5.1 Load Distribution Scenario description and derived preferences

Several sources of inspiration for the *Load Distribution Scenario* can be found such as the typical workweek with five days of work and two days off or a normal office workday with two long working periods separated by a midday break. From an abstract perspective, this scenario contains a fixed number of work phases that have to be scheduled along with recreational breaks for the worker.

Each worker might have a different preference concerning continuity and peak loads of work. A popular differentiation considers two fundamentally different expressions of working characters, which shall be detailed in the following: heavy lifters vs. slow burners, inspired by [33] .

The *heavy-lifter* type prefers longer periods of work which are very exhausting, but run without interruption. Driven by the exhaustion, longer recreational breaks are necessary afterwards. A typical arrangement of such a schedule is depicted in figure 9.

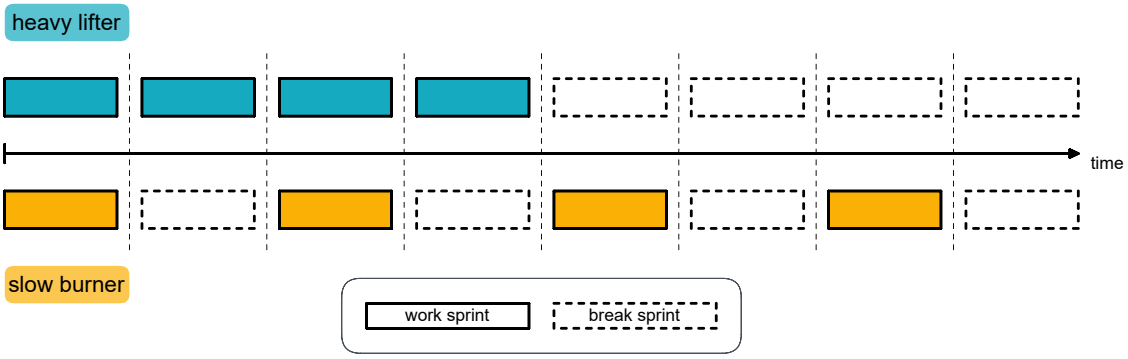

FIGURE 9: Illustration of envisioned phenotypes for concerning load distribution

The *slow burner* type prefers shorter periods of work interspersed with frequent breaks for refreshment. Since the general state of exhaustion does not reach a level as high as that of the *heavy lifter*, the duration of the breaks can be shorter, as illustrated in figure 9.

Aside from these illustrated examples, there are many configurations in between and preferences might even differ by activity-type. The current work considers only an abstract model without specific units like week-days. In the following section, a generalized formalization (loading profile) is given, which allows specifying the general ratio between work and break periods.

3.5.2 Derived objective：Deviation from loading profile (DLP)

As an objective for the abstract model, the “deviation from loading profile” (DLP) is defined. At first glance, a maximum deviation could also be considered a constraint. In this formalization it is chosen to be modelled as objective, as this allows investigating trade-offs with the fulfillment of other objectives.
The derivation of this objective is split into three stages:

1. definition of the loading profile
2. classifications according to loading profile
3. definition of a deviation measure

The current work specifies a loading profile as a recurring pattern of work and break periods. Exception rules are considered outside of scope and not realized. The pattern is described by an integer pair which is formed by the number of working phases and the desired number of adjacent break phases as indicated in figure 10.

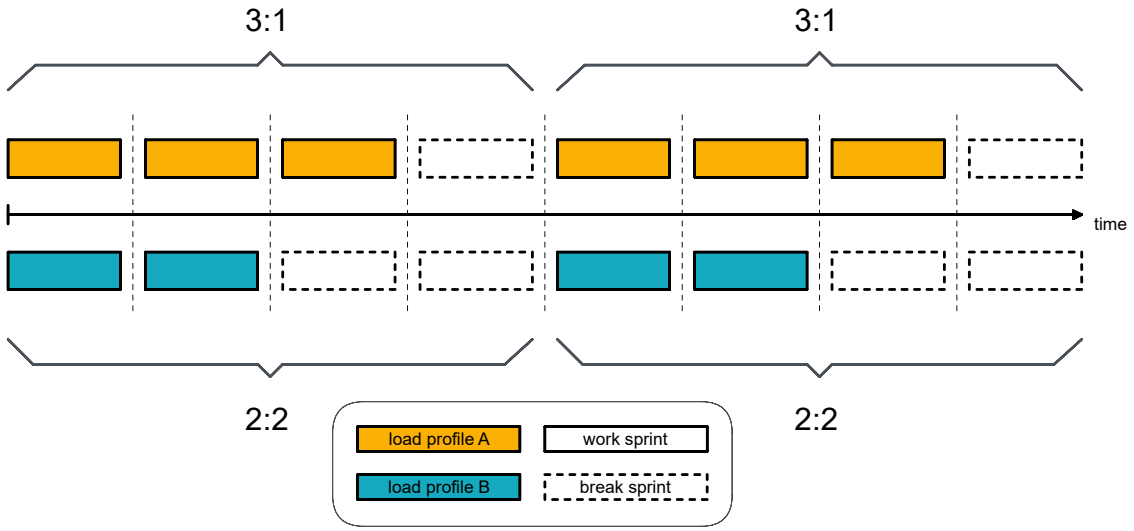

FIGURE 10： Examples of different load distribution profiles

From figure it can clearly be observed that the overall number of completed tasks within a given number of sprints depends heavily on the work:break-ratio defined in the loading profiles. This indicates that there might be a varying conflict potential with other objectives, especially with respect to the MSL objective. Variation in conflict potential is an aspect that has to be taken into account carefully during the implementation.
For objective of deviation from load profile (DLP), the observation takes place at schedule level, so the specific content of each sprint is not relevant, but it has to be known whether a sprint is classified as a work or as a break phase. In the following, every sprint that is non-empty is considered a work sprint, as defined in equation 18, 19. By

implication this also includes sprints which are violating the minimum fill grade (MIF, see section 3.4.6). This is done in order to prevent biasing the objective value towards acting as a hidden penalty term for that constraint.

$$\mathcal{S}_{\text{work, m}} = \left\{ x_{\text{SPR}} \in \mathcal{S} \mid v_{\text{fill-grade}}(x_{\text{SPR}}, m) \neq 0 \right\} \tag{18}$$

$$\mathcal{S}_{\text{break, m}} = \left\{ x_{\text{SPR}} \in \mathcal{S} \mid v_{\text{fill-grade}}(x_{\text{SPR}}, m) = 0 \right\} \tag{19}$$

The last step in formalizing the DLP objective is the definition of a comparison function which quantifies how much the schedule and loading profiles differ, so that an quantifying value can be determined. As a general approach, a windowed evaluation is chosen which is illustrated in figure 11: Deviations are measured by evaluating dynamically sized portions of the schedule instead of checking against a pre-generated, non-deviating comparison mask with the length of the schedule. With a windowed approach, an early deviation does not affect subsequent non-deviating portions of the schedule and allows rewarding longer chains of non-deviating scheduling as well as penalizing increasing chains of deviating assignments.

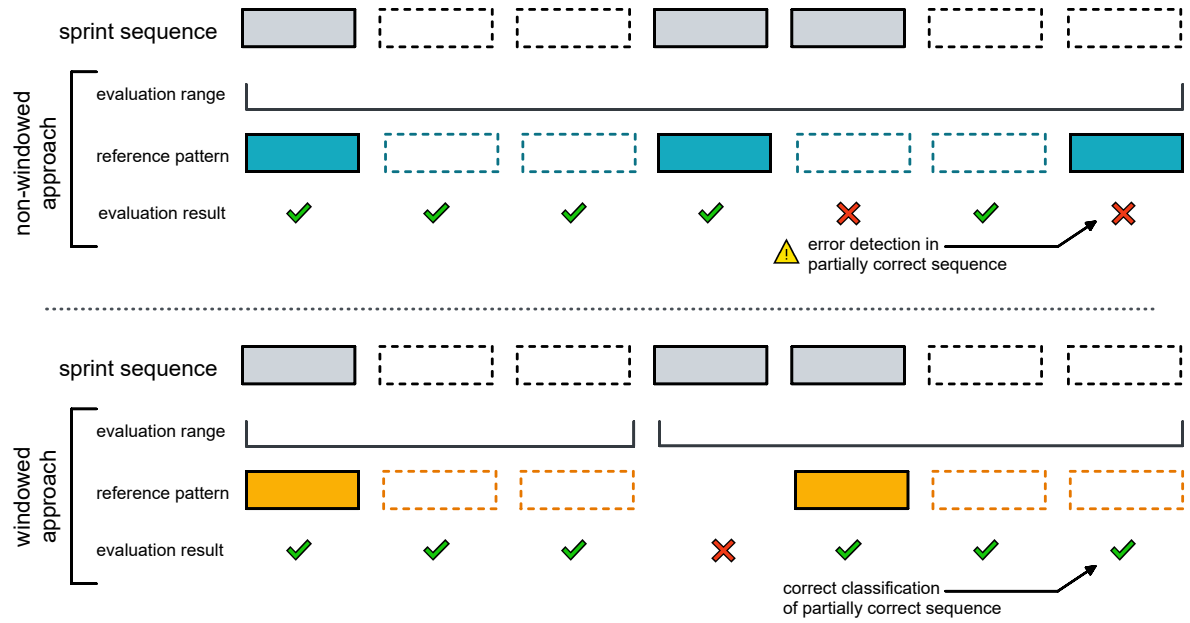

FIGURE 11: Comparison of evaluation result for non-windowed and windowed approaches

The deviation of a schedule from the specified load profile equals the sum of deviations of all windows within the schedule as shown in equation 20. One window i_{win} contains the sprints $\mathcal{S}_{i_{\text{win}}, \text{m}}$ and is formed by a full work–break alternation. Within each window the patterns (pat) of the actual schedule and of the loading profile are compared by using

the LEVENSHTEIN distance (lev) [34] which calculates the required editing distance between those patterns.

$$f_{\mathrm{DLP}}(m) = \sum_{i_{\mathrm{win}}}^{i=n_{\mathrm{win}}} \mathrm{lev}\,(\ldots \\ \mathrm{pat}(\mathcal{S}_{\mathrm{work,\,m}} \cap \mathcal{S}_{i_{\mathrm{win}},\,\mathrm{m}}, \mathcal{S}_{\mathrm{break,\,m}} \cap \mathcal{S}_{i_{\mathrm{win}},\,\mathrm{m}}), \\ \mathrm{pat}(p_{\mathrm{work:break}}) \\ \ldots) \tag{20}$$

3.6 Formalisation of Deadline Scenario

3.6.1 Deadline scenario description and derived preferences

A major element within many scheduling systems are deadlines which have to be enforced. A deadline shall be a defined as a point in time that is the latest possible finish date of an activity. For the current system model, this means that a deadline is the latest sprint acceptable for scheduling a specific activity. Tying deadlines to sprints is a rather abstract model, but it can simply be extended to a real-world calendar model by underlaying sprints with calendar dates. While mentioned here, this aspect is not considered further in this work.

Assuming a consecutive nature of all sprints in the schedule, it is possible to quantify the relation between the sprint which an activity is scheduled in and the sprint which is specified as deadline. The distance measure between these two sprints shall be labeled "deadline buffer" ($v_{\text{deadline-distance}}$), defined by the calculation in equation 21.

$$v_{\text{deadline-distance}}(x_{\mathrm{ACT}}, m) = m_{\text{deadline}}(x_{\mathrm{ACT}}) - m(x_{\mathrm{ACT}}) \tag{21}$$

A frequent preference concerning the $v_{\text{deadline-distance}}$ value is to have a reasonable buffer to a deadline. The user might have enough time to foresee problems early and can prevent stressful situations with respect to the approaching deadline. The characteristic phenotype reflecting this preference shall be labelled "early-bird finisher" and is depicted in figure 12.

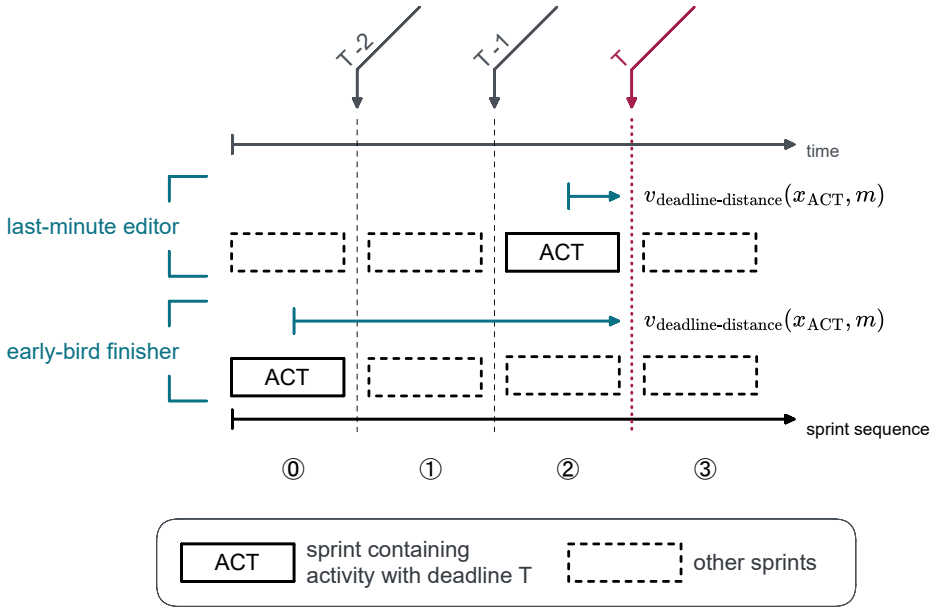

FIGURE 12: Spectrum of phenotype expressions concerning deadline scenario

According to preliminary testing, it is assumed that the maximum deadline buffer has to be limited in the current system. An unbounded value could result in an accumulation of activities with deadlines in the first sprint, which in turn would cause strong constraint violations of the maximum fill grade. For the purpose of this work, it is considered more realistic to generate an even distribution across the whole schedule.

The opposite approach shall be labelled “last-minute editor” and describes the preference of scheduling an activity as close to the deadline as possible, in the best case within the deadline sprint itself as shown in figure 12. This schedule phenotype allows to e.g. wait for last-minute changes in input from other parties and puts pressure on the user because of the approaching deadline which kickstarts productivity for some people.

3.6.2 Derived objective: Deviation from Deadline Buffer (DDB)

The following objective contains the previously described preferences as preset preferences. Preliminary test have shown that an implementation as trade-off preference does not work well, as the search space is diluted quickly. Furthermore, from a domain perspective it makes sense to collect preferences *a priori* as the user’s attitude towards the deadline buffer is assumed to have a personal standard value.

The objective shall be the minimization of the deviation between user-preferred and algorithm-generated deadline buffers. Two different types of possible deviations exist: positive (more buffer than specified) and negative (less buffer than specified). Both are treated equally, even though from a use-case perspective, deviations might have

consequences of different severity. Technically, a distinction of cases is possible, but shall not be considered here in order to prevent overcomplicating the system at an early design stage.
For the purpose of this work, a deviation metric of quadratic deviation is used. This metric is taken from the field of resource leveling and covers both cases of deviation without exceptions [35]. The resulting objective shall be labelled f_{DDB} and is defined in equation 22.

$$f_{\mathrm{DDB}}(m) = \sum_{x_{\mathrm{ACT;\,i}} \in \mathcal{A}} \left(v_{\text{deadline-distance}}(x_{\mathrm{ACT;\,i}}, m) - p_{\text{deadline-buffer}} \right)^2 \tag{22}$$

3.6.3 Derived constraint: No Deadline Failure (NDF)

Introducing deadlines to the system model implies that there is the possibility of activities scheduled after the specified deadline in which case the deadline is missed. In return, this means that the full time span prior to the deadline forms valid scheduling slots. Such feasible regions can be modelled via an indicator function $e_{\text{deadline-fail}}$ (see equation 23) and the resulting equality constraint is specified in equation 24.

$$e_{\text{deadline-fail}}(x_{\mathrm{ACT}}, m) = \begin{cases} 1 & , m(x_{\mathrm{ACT}}) > m_{\text{deadline}}(x_{\mathrm{ACT}}) \\ 0 & , \text{otherwise} \end{cases} \tag{23}$$

$$h_{\mathrm{NDF}}(m): \;\; 0 = \sum_{x_{\mathrm{ACT;\,i}} \in \mathcal{A}} e_{\text{deadline-fail}}(x_{\mathrm{ACT,i}}, m) \tag{24}$$

In the currently considered system, it is assumed that deadlines must not be missed. This prerequisite is motivated by the fact that deadlines are often set by external factors that might not be negotiable by the user. Under this assumption, a solution which contains deadline violations is considered as an inviable solution and therefore a strict constraint handling is required.

3.7 Formalisation of Robustness vs. Performance Scenario

3.7.1 Robustness vs. Performance scenario description and derived preferences

The last scenario shall introduce a further real-world perspective: There is a natural error in estimating activity durations. Some scheduling methods like PERT account for this by requiring the user to specify a worst, best and average case (see section 2.1). While this approach leads to a high safety factor of the schedule, it may also be perceived as tedious and requires extensive experience.

As a simplified measure against schedule slips, a buffer capacity within sprints shall be introduced for the currently investigated system. This means that activities maintain the specified fixed duration, but users provide a preference for how much capacity within a sprint shall be left unscheduled for covering potential estimation errors. Two types of typical phenotypes can be imagined for this preference: Higher-Performance-Lower-Robustness (HPLR) vs. Lower-Performance-Higher-Robustness (LPHR).

The HPLR phenotype is illustrated in figure 13: There is no buffer planned, so that the schedule can complete within a minimum schedule length using the minimum number of sprints. However, this approach risks breaking the whole schedule in case of activities which exceed the planned duration.

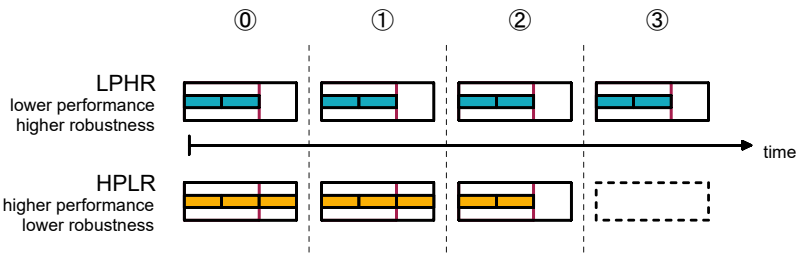

FIGURE 13: Examples of phenotypes for HPLR and LPHR preference

The second phenotype (LPHR) incorporates additional, empty capacity within each sprint as indicated in figure 13. This allows deviation in the actual activity duration preventing a slip of the entire schedule at the expense of a longer overall schedule.

3.7.2 Derived objective: Deviation from Target Buffer (DTB)

For the same reasons as the deviation from deadline buffer, the deviation from target buffer (DTB) shall feature a preset preference. This value is defined as $p_{\text{target-buffer}}$ and is

collected *a priori*, accordingly.
In contrast to the DDB objective, the current observation model is the individual sprint. A metric for this objective has to compare the actual fill grade against the preference value for each sprint and has to quantify a deviation value for the whole schedule. Generating a solution without deviation is rather unlikely, therefore this constraint shall be formalized as inequality. Applying a quadratic deviation metric results in the objective defined in equation 25.

$$f_{\mathrm{DTB}}(m) = \sum_{\substack{x_{\mathrm{SPR;\,i}} \in \\ \{x_{\mathrm{SPR}} \mid \ 0 \\ = e_{\mathrm{overload}}(x_{\mathrm{SPR}}, m) \\ = e_{\mathrm{underload}}(x_{\mathrm{SPR}}, m)\}}} \left(v_{\text{fill-grade}}(x_{\mathrm{SPR;\,i}}, m) - p_{\text{target-buffer}} \right)^2 \tag{25}$$

For the special case of $p_{\text{traget-buffer}} = 0$, the optimal phenotype might have features similar to the optimal utilization of sprints which is already formalized in f_{MSC}. This points to a varying conflict potential of these objectives depending on preset preferences, which shall be investigated in the experimental evaluation.

4. Implementation of Evolutionary Multi-Objective Activity Scheduling

This chapter focuses on the description and justification of implementation decisions made to realize the proposed method using a genetic algorithm.

4.1 NSGA-II algorithm as selected metaheuristic

In the field of MOO, there are numerous different state-of-the-art algorithms available. Among the best known is the *non-dominated sorting genetic algorithm II* NSGA-II. This algorithm extends the concept of genetic algorithms not only by respecting *pareto dominance*, but also by establishing a rank-based sorting procedure for non-dominated solutions as well as a *crowding distance* measure in objective space. By applying these procedures, the results in many test problems are very good [36]. An extended variant NSGA-III has been developed which is superior at optimizing higher numbers of objectives, but requires some kind of normalization by design [37]. Due to the relatively few information concerning the objective space of the given problem, a normalization is no easy undertaking. In consequence, NSGA-II is chosen for implementation over NSGA-III as no normalization is needed by default.

4.2 Design decisions regarding problem encoding

Selecting an encoding of the chosen formalization means considering all potential system elements. For each element, a decision has to be made how the algorithm perceives and interacts with it. In case of the given problem, relevant elements or aspects are:

- ① general mapping function which assigns activities to sprints
- ② interaction of the capacity and its fill grade of a single sprint
- ③ relation between the scheduled position of an activity and its deadline

Aspects ② and ③ are highly dependent on ① because of the selected formalization. Dependent aspects are investigated first, as these might contain requirements relevant for encoding aspect ①. In general, it is difficult to find a direct encoding for highly constrained problems like scheduling [10]. Observations concerning aspect ② (mapping function) of the given problem happen at sprint level and require an aggregation function. The aggregation equals a decoder function, so it is not possible to realize a direct encoding of aspect ② (mapping function) with respect to the chosen formalization. For aspect ③ (positional relation) a comparison function is needed which checks the dynamic (scheduled) value and the fixed (deadline limit) value. Representing two different states in

a direct encoding is difficult, and consequently aspect ③ (positional relation) also motivates an implementation via an indirect encoding.

Based on the decision to realize aspects ② and ③ via decoder functions, the core aspect of mapping activities to sprints (①) can be treated in isolation. This aspect is analog to the mapping performed for the standard BPP, so established encodings can be found in the literature. The mapping itself is a two-dimensional problem where m activities have to be assigned to n sprints. In linear programming this can be represented by a binary-valued vector for each activity [17]. Implementations for meta-heuristics propose e.g. integer string or matrix-based encodings [23].

For the current problem, an integer vector is used as a base encoding, which maps an activity to its designated sprint as shown in figure 14. This encoding forms a *one-to-many* cardinality, which enforces the UQA constraint by default. No computationally costly checks are required, as each activity occurs only once in the vector. As a result, the search space is permanently mapped to feasible regions only and consequently respects the *viable at any time* requirement for this constraint.

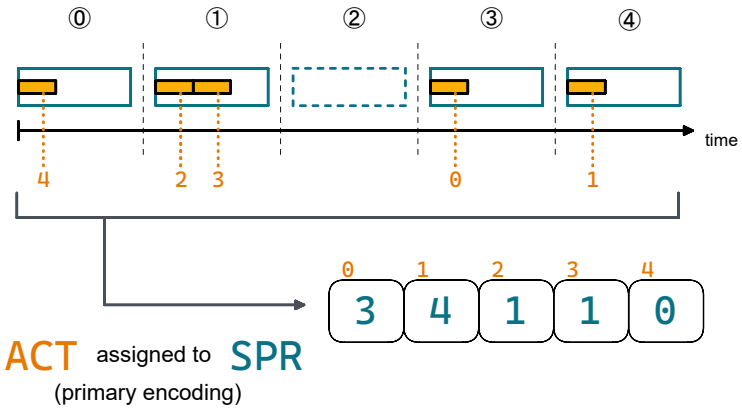

FIGURE 14: Integer-vector-based encoding for schedule

A sequential, vector-based encoding is close to DNA sequences, which inspired the basic operators used in genetic algorithms [25] . This implies that vector-based encodings are generally well-suited for applying standard operators. For the given problem, this is only true for cross-over operators. For mutation operators, this is not the case because the given problem features multiple observation levels. Some operations at schedule level, e.g. swapping the content of two sprints, would require numerous, well-coordinated mutation steps.

Multiple mutation steps are an unfavorable design concerning the *proximity* requirement for operators, which states that closely related phenotypes shall also be close to each other in genotype space and therefore easily reachable for search operators [10].

In order to overcome these limitations, a hybrid encoding shall be introduced here which means that a supplementary encoding for some operations is created. The secondary encoding shall adhere to the *proximity* requirement and benefit operations at schedule level. Such an approach has

analogies with the observer pattern [38] which is a common design blueprint in computer science and provides different views onto the same data.

In the given case, the first encoding depicts the schedule at activity level via a vector, while the secondary one depicts the schedule at sprint level via a matrix indicated in figure 15. The envisaged hybrid encoding thus combines two common encodings from the literature proposed for meta-heuristics. It has to be noted that the second encoding does not enforce UQA feasibility. Consequently, this limitation has to be considered during the design of all operators it is applied to.

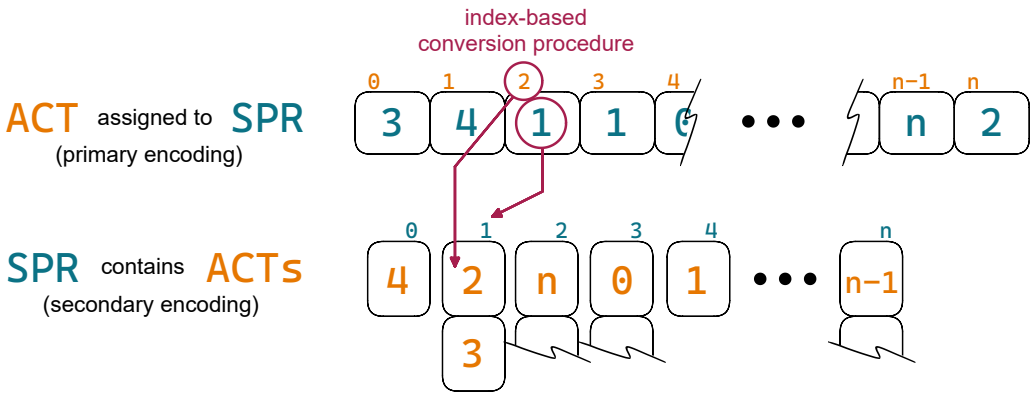

FIGURE 15: Hybrid encoding with conversion between vector- and matrix-based representation

Introducing additional computational steps generally has to be investigated carefully concerning performance. The conversion process between two different encodings is computationally expensive because it has to be executed at each generation for each individual bidirectionally. The conversion itself, however, might be coded quite efficiently when all representations are built around indexed data structures. The iteration of indices means that the operation has a deterministic complexity. Therefore, it is assumed that the benefits of a conversion outweigh the additional effort, especially with respect to the alternative procedure (a non-deterministic search with a non-proximity-optimised operator).

4.3 Design decisions regarding sampling operation

Infeasible solution can be injected to the population even in the very first step of the optimization process: the initial sampling of individuals. Additionally, disadvantageous distributions of data points generated during sampling restrict the genetic diversity of the population or bias it towards unfavorable regions [9]. Consequently, the selected sampling method as well as range of potential values is essential for designing an effective optimization approach, especially concerning the *viable at any time* requirement.

In the literature, multiple sampling methods can be found for retrieving an advantageous distribution of data points for the initial population. Aside from these general approaches, there are also specialized approaches for standard problems like BPP, which for example feature using a specific heuristic for the initial sampling and optimizing the decision variables afterwards via evolutionary computation [17].

Due to the newly introduced constraints, such existing specialized approaches are not considered applicable, so the sampling shall be done via an existing more general sampling method. The authors of [39] illustrate that different sampling methods not only have a significant influence on convergence behavior, but are also of varying effectiveness for different test problems. Consequently, a well-informed decision results in better-performing algorithms. As there exists no prior knowledge of the exact shape of the fitness landscape here, a generic random sampling is chosen as a baseline.

Sampling, in correspondence with encoding, forms an effective approach to map the search space, i.e. available genotypes, to desired regions only. This mapping however only make sense for constraints that can be checked against static bounds and for those without aggregation functions. In general, aggregation functions encapsulate a combinatorial aspect, which means that their evaluation already spans an own small search space of potentially high computational complexity. This implies that trying to eliminate these inviable solutions a priori will further complicate the problem.

Resulting from these considerations, it has to be decided which constraint violations can be prevented via search-space mapping.

In the given problem, four different constraints exist. As described in section 2.3 overly strict constraint handling creates a strong evolutional pressure which leads to a rapid extinction of the population. This implies that a strict, rejection-based handling shall only be applied to constraints that absolutely require it. In the current work, this necessity is depicted by the *viable at any time* requirement that holds for three of the constraints:

- UQA the feasibility of which is enforced by default as long as the sampling happens in primary encoding as explained in section 4.2.
- MAF, which requires checking the current fill grade against a static maximum value. Calculating the fill grade contains an aggregation function, making it ineligible for a search-space mapping under the given considerations. Other constraint-handling techniques have to be considered.
- NDF, which is checked against a static bound in search space via a simple, non-aggregating decoder function. Consequently, search-space mapping is applicable here.

The chosen primary encoding features a fixed mapping of activities to a gene in the chromosome vector. The allele might change, but the gene is not repositioned. This fixed mapping allows using

a predefined range of possible values for each gene. In the given formalization, the value of a gene corresponds to the currently scheduled sprint. Consequently, NDF feasibility can be reached a priori by limiting possible sampling values to sprint indices less than or equal to the deadline value of activities, as indicated in figure 16.

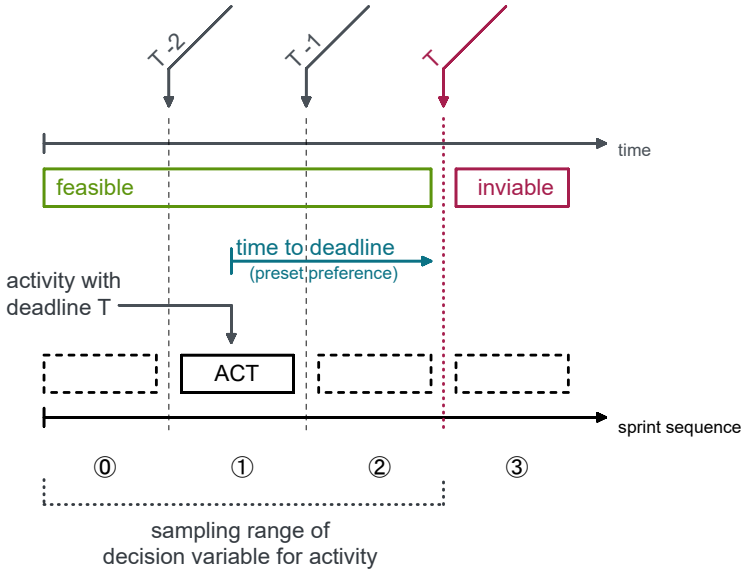

FIGURE 16: Limiting sampling values for enforcement of deadline feasibility

Having a feasible initial population does not guarantee that the feasibility persists during evolution. Operators which do not respect all semantics of the encoding by default might generate new infeasibilities. In consequence, every operator has to checked whether the semantics of the encodings and the bounds of the sampling remain intact.

4.4 Design decisions regarding crossover operators

As introduced in section 2.6, cross-over operators imitate DNA recombination. The presented hybrid encoding provides two different representations for applying genetic operators. In the following, only the primary encoding shall be considered for cross-over operations, which means that the recombination is run in an activity-based representation. All following considerations only take standard cross-over operators into account.
Using the primary encoding only is motivated by two reasons:

1. UQA feasibility is maintained by default in contrast to the secondary encoding.
2. Recombined solutions in sprint-based representation are assumed to be a sub-set of the solution set that can be obtained from activity-based representation because the secondary encoding is a grouped representation of the same data.

This preliminary consideration, however, has not been backed by mathematical proof yet. Attempts for experimental evaluation of the performance impact originating from different representations can be found in the literature concerning a related field, but cannot provide a decisive result [40].
The application of genetic operators might influence the stability of feasibility, which describes whether new infeasibilites are introduced to formerly feasible solutions or not. The "viable at any time" constraints which have to be checked concerning this matter are NDF and MAF. With respect to the fixed position of a gene within the chromosome vector, it can be deduced that NDF feasible mappings of activities to sprints remain intact, as does the search-space mapping. MAF constraint violations can however only be detected in phenotype, but the cross-over mechanism operates in genotype space. Missing a decoding, standard cross-over operations are very likely to produce inviable solutions and a constraint-handling mechanism has to be introduced for MAF.
The given problem has no fixed size of the individual problem instance, which means that the operator selection has to be validated against the influences of different data-set sizes. Smaller problem instances result in a shorter chromosome vector, while larger instances extend the length of the vector. A well-chosen cross-over operator has to be independent of the problem size.
Not all classic operators are independent of the problem instance. Operators with a fixed number of cross-over occurrences like 1-point-CX lose influence relative to chromosome length. For this reason, uniform cross-over UX is selected for this implementation which has generally a good performance for many problems [10].

4.5 Design decisions regarding mutation operators

High-dimensional combinatorial problem can have an extremely large, scattered search space with disconnected regions. As explained in the introduction (see 2.6), mutation provides a means to explore distributed regions in search space and diversify values of decision variable beyond the sampled values.
It is not uncommon to apply multiple mutation operators in a genetic algorithm [41]. This implementation shall feature more than one operator because the formulation contains different objectives which have varying observation levels. Depending on objective and observation level, different operators can have a different influence.
These variant-rich circumstances open a plethora of potential approaches towards mutation operators. Multiple candidates have undergone preliminary investigations concerning their contribution to improving the optimization results. By envisioning the operating principles of mutation approaches, these can be broken down to atomic components: E.g., the process of swapping two activities between two sprints equals two independent activity moves in a row which are applied to the same sprints. However, from a statistical perspective, a full swap via independent moves is far less likely to occur than a planned swap by a combined operator.
The effects of combined operators form an extensive subject on its own that is undergoing research. Analyzing their exact interaction requires extensive statistical experiments [42,43].

Combined operators therefore introduce a new design dimension. In order to reduce hidden interactions as much as possible, this implementation is built around two atomic operators which represent two common concepts in combinatorial problems [41]: exchange and rearrangement.
Of these two selected mutation operators, which are executed on randomly selected genes, one works at activity level while the other works at schedule level:

1. *activity-shifting* mutation ≙ moving a selected activity from one sprint to another
2. *sprint-swapping* mutation ≙ interchanging the position of two sprints (including all assigned activities) within the schedule

Both operators are designed independently of MAF-feasibility considerations, as none of the previous operators can guarantee to deliver feasible solutions. Consequently, constraint violations are treated by constraint-handling methods at the end in order to avoid multiplication of computational effort. A *viable at any time* implementation without dedicated constraint would have to incorporate feasibility checks in the current algorithm step.
A mutation at activity level is required to generate new values of decision variables which have not been sampled. The activity-based representation (primary encoding) is used here because it is the straightforward option from a technical as well as domain perspective and leverages the bounds of decision variables for ensuring NDF feasibility. In order to mutate the chromosome, randomly selected decision variables are resampled within their defined value range.
Changes at activity level are not directly relevant for objectives like MSL or DLP, as these are observed at schedule level, where it is only relevant whether a sprint is loaded or not. Following from this, moving individual activities has only limited influence on these objectives. This assumption motivates designing a mutation which utilizes the sprint-based representation.
The selected *sprint-swapping* mutation is a bidirectional operation in contrast to the *activity-shifting* mutation. Shifting the position of a single sprint would cascade changes in the position of all sprints following the position. After preliminary considerations, a *sprint-shifting* mutation is considered too close to a random search.
Since a sprint-level mutation operator does not affect the individual sprint composition, it neither affects MAF nor MIF constraints. Nevertheless, it shifts activities in time, so the NDF constraint might pose a design challenge. Due to careful design decisions concerning the preceding algorithm steps, the solution is assumed to be in an NDF-feasible state before the sprint-level mutation. Consequently, moving a sprint to an earlier point in time will not generate constraint violating solutions because the activities are moved away from potential deadlines. In contrast, moving a sprint forward in time might exceed a deadline of one activity in the sprint. To eliminate this possibility, a feasibility-aware operator shall ensure that a sprint is not shifted forward beyond the earliest deadline of one of the contained activities. The resulting operator is described in algorithm 1.

ALGORITHM 1: Deadline-viability-aware mutation using sprint-level

SprintSwappingMutation(x_{SCH})

in: x_{SCH} schedule in sprint-based representation
out: x'_{SCH} mutated schedule in sprint-based representation

begin

$i_{\text{fwd-swap}} \leftarrow \text{random}(1, n_{\text{SPR}})$

$x_{\text{SPR, fwd-swap}} \leftarrow x_{\text{SPR, i}} \in x_{\text{SCH}}$ with $i = i_{\text{fwd-swap}}$

if $m_{\text{deadline}}(x_{\text{ACT}}) = \emptyset \ \forall x_{\text{ACT}}$ in $x_{\text{SPR, fwd-swap}}$ **then**

  $i_{\text{fwd-swap, max}} \leftarrow n_{\text{SPR}}$

else

  $i_{\text{fwd-swap, max}} \leftarrow \min \{m_{\text{deadline}}(x_{\text{ACT}}) | \ \forall x_{\text{ACT}}$ in $x_{\text{SPR, fwd-swap}}\}$ ▷ viabilit

end

$i_{\text{bwd-swap}} \leftarrow \text{random}(i_{\text{fwd-swap}}, \ i_{\text{fwd-swap, max}})$

$x_{\text{SPR, bwd-sprint}} \leftarrow x_{\text{SPR, i}} \in x_{\text{SCH}}$ with $i = i_{\text{bwd-swap}}$

$\text{swap}(x_{\text{SPR, fwd-swap}}, \ x_{\text{SPR, bwd-swap}})$

end

4.6 Design decisions towards constraint handling

The previous implementation decisions do not cover handling violations of the MAF and MIF-constraints, yet. As the operators cannot guarantee feasibility for these constraints, dedicated handling mechanisms have to be implemented. Both constraints are considered as a *representation decoding constraint* (in analogy to the *representation decoding objective* defined in [10]), which means that the function value cannot be determined from genotype via a simple look-up. In contrast to the NDF constraint, an aggregating decoder function is needed here.

The MIF-constraint is not required to be *viable at any time*. Consequently, an implementation can also consider handling approaches which tolerate infeasible solutions. In the formalization, this constraint was already relaxed to an inequality constraint, see section 3.4.6. For a well-founded decision, it has to be assessed whether the tolerances introduced by relaxation provide sufficient solutions when combined with a rejection-approach.

In general, a high value for the threshold values ($v_{\mathrm{th,\,USC}}$, $v_{\mathrm{th,\,UAA}}$) limits the final usability of the solution, smaller values however results in a higher rejection rate. Especially, in the initial population and first phase of optimization many constraint violations are expected, as activities are sparsely distributed across all available sprints. This behavior contradicts the idea of supporting exploration in the initial optimization phase, as the evolutionary pressure is already quite high. Resulting from these considerations, a fixed threshold value is not regarded as a well-performing approach.

A common approach for controlled toleration of infeasible solutions is biasing the fitness via a penalty function [19]. Applied to multi-objective use-cases, this translates to a custom penalty for each objective [23]. Such an approach is difficult to implement because it requires some kind of normalization or weighting factor, so that the solutions are biased predictably. Other methods like constraint-domination [23] can have the same short-comings when used without normalization. Lacking knowledge on the fitness landscape, a proper normalization is a difficult undertaking.

Concluding, a method for dynamic constraint handling is required that is not tied to the objective values and can be implemented without normalization.

Therefore, this work proposes a time-dependent constraint-handling method labeled *shrinking living space* method. This method is based on gradually decreasing the threshold-values $v_{\mathrm{th,USC}}$ and $v_{\mathrm{th,UAA}}$ in order to increase evolutionary pressure during the optimization, as shown in figure 17. Under the assumption that a fitness landscape contains disconnected regions which are separated by constraints, it is hoped that this method allows parts of the population to bridge these gaps during the early generations. The approach has some similarities with the mentioned *objective rescaling technique* (see section 2.3), however, this method is rather a *constraint rescaling* function. There are also some overlaps with *successive bound reduction* found in numeric optimization [44].

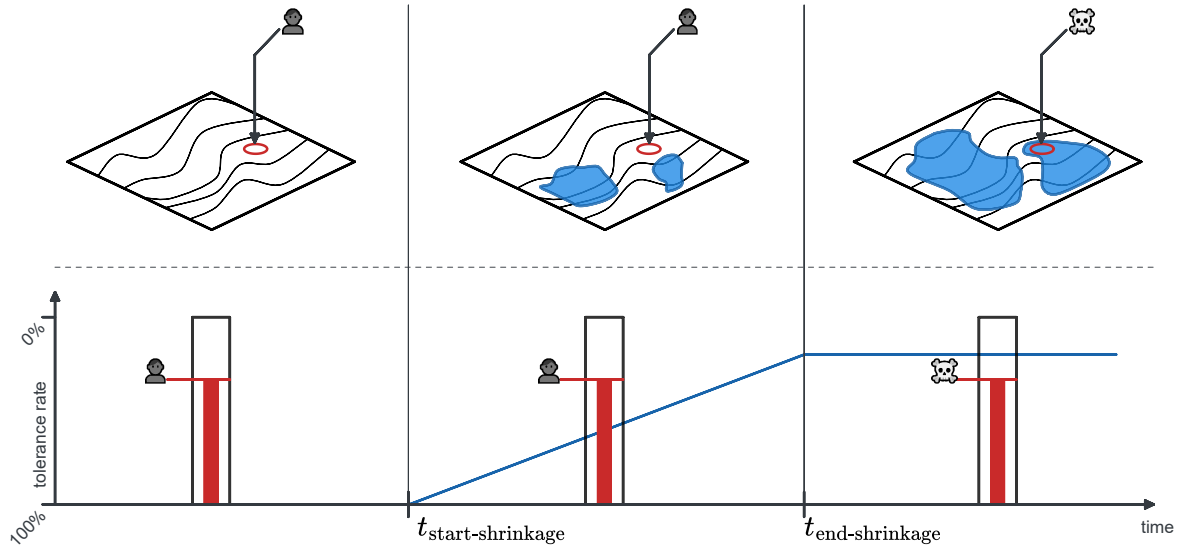

FIGURE 17: Illustration of the envisioned model of a shrinking living space

Concerning the MAF constraint, solutions were required to be *viable at any time* according to the initial system model specifications, i.e. there must not be a constraint violation at any time. Since the operators introduced so far, do not take this into account a non-rejecting repair would be required for a proper solution that satisfies all system requirements. The naive approach would be re-scheduling the fraction of activities, which cause an overload to the first sprint with remaining capacity. This procedure is comparable to the known *first-fit decreasing* algorithm [17], an established specific heuristic for BPP. The general applicability is limited, since in worst case an edge case is created where no NDF-feasible sprint is available. As a result, space would have to be created in an already occupied sprint triggering a cascading reschedule.

Covering such edge cases within a repair operator points strongly into the direction of developing a dedicated specific heuristic, which is beyond the scope of this work. In consequence, the *viable at any time* requirement is revoked for the MAF constraint and the constraint handling is left to evolutional mechanism, i.e. assigning a lower rank to infeasible solutions during selection, which eventually leads to extinction of these phenotypes.

5. Experimental evaluation

5.1 Experimental tools

5.1.1 BBP benchmark suite by SCHOLL

Many common benchmark data sets for scheduling problems are built around sequence information [45], so that sets of interrelated activities or jobs are scheduled. This foundation restricts the applicability in the present work, as sequence information is intentionally left out. Currently, creating a full custom benchmark is out of scope, so the chosen approach is to use an existing benchmark and apply necessary modifications.
As described in the formalization the current problem is closely related to the original BPP. This similarity makes benchmarks which have been created for the BPP well-suited as a basis for creating test data for the current system model. From multiple available data sets, the one created by SCHOLL is picked. Each instance of the benchmark specifies a bin capacity as well as a fixed number of items, characterized by an item weight randomly chosen from a given interval. Additionally, the number of bins required for an optimal packing is provided [46].
For application to the given problem, the item weight is considered as activity duration and the bin capacity is used as sprint capacity.

5.1.2 Modifications to the benchmark data set

The introduced benchmark dataset was developed for the BBP, thus not all system elements of the given problem are depicted, e.g. deadlines. Consequently, the data set has to be adapted and extended.
While the original BPP definition does not specify an upper limit of bins available, a real-world scheduling problem is often built around a certain planning horizon, which is also a key element in the proposed scheduling framework (see section 3.2). The upper limit of sprints available was set to twice the total number of activities to schedule, based on the idea that the design space should not be too constrained with respect to the additional objectives (f_{DTB}, f_{DLP}).
As explained in the scenario formalization (section 3.6), the deadline for an activity is modelled by the index of the last sprint allowed for scheduling. The actual deadline values are generated by a random sampling of sprint indices, which are randomly assigned to activities. The more specified deadlines a problem contains, the harder it becomes to solve, as each deadline constrains the solution space further. Therefore, the overall complexity shall be limited by adding deadlines to only one third of the activities.

Two potential issues of these adjustments can be identified beforehand, which could generate distortions of the experimental results: Firstly, deadline values at the beginning of the schedule could interfere with the f_{DDB} objective and secondly, multiple sprints with the same deadline index can cause violations of the h_{MAF} constraint.
The chosen system formalization allows specifying a buffer amount for f_{DDB} via a preset preference. If a deadline index is smaller than the specified buffer, the resulting objective value has an inevitable negative bias. In a production-ready system, such a bias may be mitigated by basing the objective dynamically on the available buffer instead of a fixed value. The current implementation acts only as proof-of-principle, so in a pragmatic approach, the range of available deadline indices is limited to values larger than the largest possible buffer preset.
The second possibility of distorted results arises when multiple activities have the same deadline value, but the total duration of those activities exceeds a sprint's capacity. A schedule without violating the h_{MAF} constraint or negatively biasing the f_{DDB} objective is not possible. This type of scheduling conflict is common in real-world use cases, but limits the comparability of experimental results in a controlled environment. Therefore, the deadline sampling operation is configured so that each index may be used once only.

5.1.3 Pymoo as selected optimization framework

All implementations concerning system formalization, algorithm design and test bench were done in Python and with the help of the optimization library pymoo. This framework focuses on providing an easy-to-use interface for running multi-objective optimization with state-of-the-art algorithms [47]. Pymoo was chosen because of its python code base, open-source approach and customization interface. Custom operators may be created using predefined slots and even core components can be altered due to the open-source code base.
While the custom genetic operators as well as system elements were implemented using the standard customization slots, a configuration change at the core components had to be made to enable testing the concept of a shrinking living space. In the default configuration, individuals are only evaluated upon injection into the population, allowing a less costly computation. The concept of a shrinking living space adjusts the constraint function, which implies that the returned value is generation-dependent. The main program loop had to be customized so that already existing individuals are re-evaluated at every generation including an update to their constraint violation values. This procedure brings substantial computational overhead, which is accepted for experimental testing and discussed in the evaluation with respect to production-level use.

5.1.4 System and algorithm configurations under investigation

The starting point for experimental design are the intended experimental results. The current system model as well as algorithm components are at a rather early design stage. Experiments are therefore meant to act as a filter for this initial design exploration, so that a future design direction can be indicated. Incorporating real-world data and fine-tuning hyperparameters is not covered in this work.

There are two major criteria from a design perspective: It has to be investigated whether the created solution solves the right problem (system formalization) and if the solution is the right tool for the problem (algorithm configuration). Both the system formalization and algorithm configuration of the current work feature a variant space of configurations. The selected configurations under investigation are detailed in the following.

Each experiment configuration is supposed to contain all objectives and constraints. Variation is introduced by the preset preferences encoded within some objectives. Users are able to select their desired preference from a closed interval of possible values, which however implies that there is a potentially infinite number of possible combinations. The initial experiments conducted in this work shall validate basic assumptions only. Consequently, only extreme or typical values are considered for preset preferences. In figure 18 all system configurations under investigation are shown in the form of a variant graph, a representation typically used for describing combinatorial variants in genomics [48].

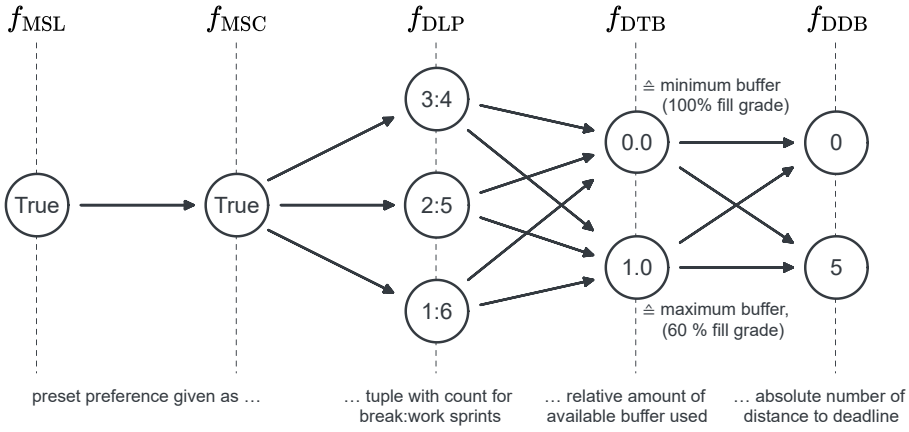

FIGURE 18: System configuration space (objective inclusion and preset preference values) depicted as variant graph

Experiment series based on evolutionary computation often feature testing different algorithm configurations in the form of varying hyperparameter settings. In this work, the main focus however lies on investigating the effect of newly designed algorithm components. Consequently, the hyperparameters are set to standard values obtained from the literature as shown in table 1

TABLE 1: Hyperparameter values for experimental series

hyperparameter	value
termination generation*	200
population size [49]	100
number of offspring per generations	100
crossover rate [36]	0.9
activity reschedule mutation rate**	0.1
sprint swap mutation rate**	0.1

*based on preliminary convergence studies using running metric [50]

**motivated by similarities to multi-dimensional BPP at an abstract level [49]

Even though constraints are technically part of the system model, the concept of shrinking the living space is considered as part of the algorithm design. Therefore, this concept is the second algorithm component to be investigated along with the custom sprint swap mutation operator. Any combination of these approaches and the standard implementation shall be part of the experimental series. The resulting configuration space is shown in figure 19 accompanied by the configuration labels that will be used later in the evaluation to refer to these configurations.

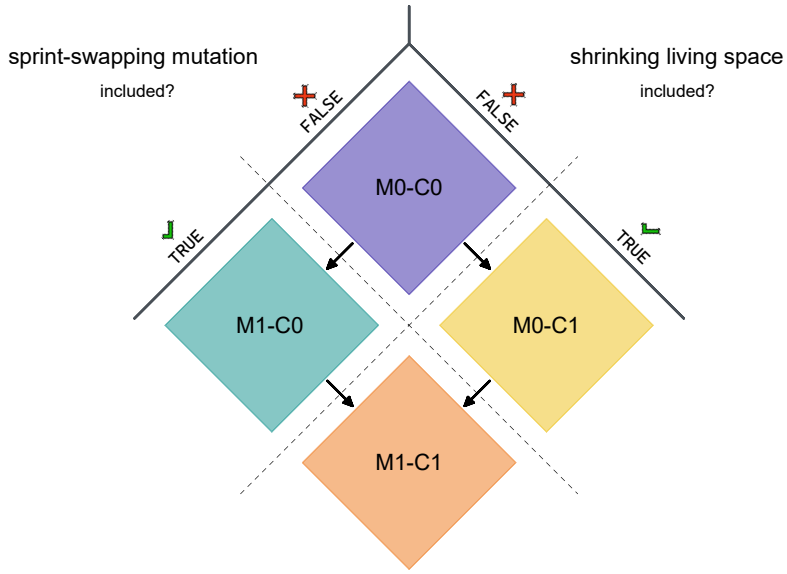

FIGURE 19: Algorithm configuration variants for testing custom operators (Associated colors mark respective variant in the following.)

For the experimental runs, each system configuration is combined with each algorithm configuration, resulting in 48 different experimental configurations. The full set of configurations is tested on an instance from the adjusted benchmark data set, configured according to table 2.

TABLE 2: Configuration parameters of problem instance

parameter	value
n_{ACT}	50
c_{SPR}	100
$n_{\mathrm{SPR,\,max}}$	100
w_{ACT}	[1, 100]

Lastly, the stochastic nature of evolutionary computation has a considerable influence on the outcome of the optimization. In order to prevent false conclusions because of misleading results originating from random effects, each experiment instance which shall be analyzed quantitatively is executed 31 times, so that significance and median can be determined with ease [51].

5.2 Metrics for evaluating algorithm as well as system design

For evaluating the experimental outcomes, four main metrics are used: objective correlation heatmap, pareto front contribution rate, normalized hypervolume and moment of inertia of the population. The motivation for this tool set as well as the selected points of interests for evaluation are introduced in the next paragraphs, while details on applying these metrics are given in the subsequent sections.

A general challenge in comparing different algorithm runs and configurations is to establish a basis for comparison between potentially fundamentally different conditions. Commonly, the pareto front is used as a reference for fair comparisons. Since these optimal solutions are unknown for the current problem, a surrogate pareto front (section 5.2.1) is established as a reference before applying other metrics.

The given system model was newly created and leverages untested approaches like preset preferences. The main target of this MOP approach is to provide not simply an optimal solution, but a set of diverse options with different trade-offs. An evaluation at system level shall verify the general design pattern and test whether the test bench results stand up to use-case expectations. A key interest is the investigation of conflict potential between objectives, as well as potential influencing factors like algorithm configurations and preferences. These conflict potential are evaluated using a correlation heatmap, as detailed in section 5.2.2.

An aggregation of all experiment runs shall form a basis for first evaluations at algorithm level. With the aid of the global surrogate front, different algorithm configurations are compared in terms of general explorative capabilities and the range of retrieved objective values.
For an assessment of the practical applicability of the current solution, special attention is paid to the average performance of the algorithm. Indicators of performance are calculated at the level of individual experimental runs and the distribution of results is examined. Diversity and quality of solutions in objective space are measured by a normalized version of the well-established hypervolume metric, detailed in section 5.2.1. The given problem is multi-modal, which means that different genotypes can produce the same values in objective space [21]. Therefore, diversity in decision and objective space have to analyzed separately, especially with respect to the concept of a shrinking living space. An moment-of-inertia-based metric for genotypic diversity is used as explained in section 5.2.4. Both metrics are also used to conduct a basic sensitivity analysis focused on different algorithm configurations and the operators used.
For evaluating the results concerning their statistical relevance, the WELCH's t-test has been selected, based on the finding of [51]. The statistical significance is denoted by a "*" according to the standardized notation from [52] as following: $p < 0.05$ ➔ *, $p < 0.01$ ➔ ** and $p < 0.001$ ➔ ***

5.2.5 Deriving a surrogate pareto front

For the given problem, the true pareto front is unknown. For such cases, [53] proposes creating a surrogate front, which is a front of non-dominated solutions obtained from the individual non-dominated sets of solutions of all algorithm runs under investigation. This surrogate front can serve for comparing these algorithms on a shared basis as shown in [50].
In order to derive such a surrogate front, all sets of non-dominated solutions are combined and then filtered by domination, so that all dominated solutions are removed from the surrogate set. As indicated in figure 20, from this surrogate front, global ideal point (best values for all objectives [21]) and nadir point (worst values for all objectives [21]) can be inferred, which allow normalization as done in section 5.2.3. These composite points are free from bias of local non-dominated solutions which are dominated globally.

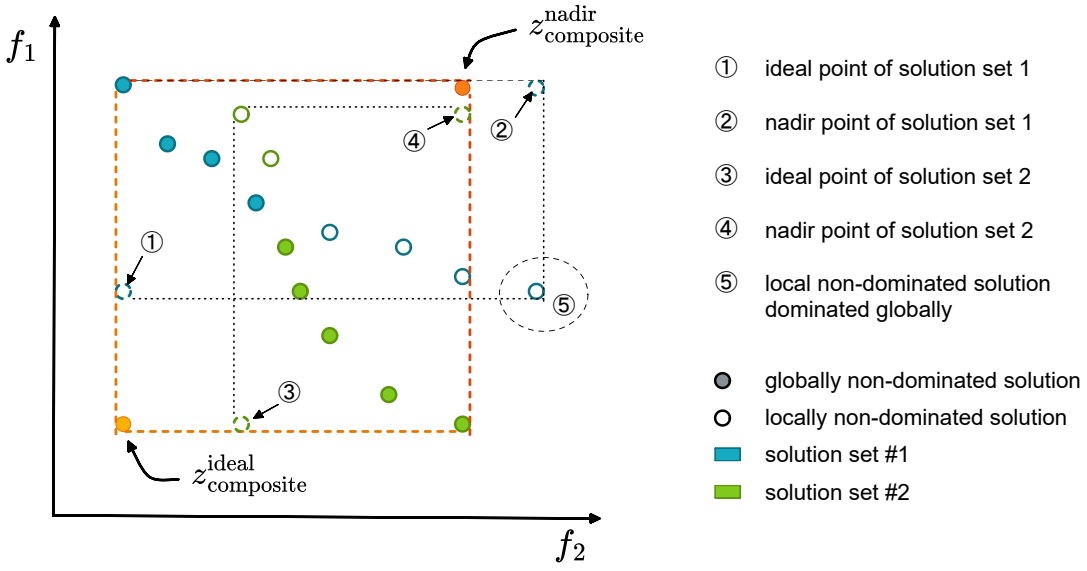

FIGURE 20: Surrogate front as well as composite ideal and nadir points from two solution sets

5.2.6 Correlation heatmap as a conflict measure

The conflicting nature of objectives between two objectives can be considered as a negative correlation. A negative correlation is given when improving one the value of one objective worsens the value of another objective. Consequently, the correlation of two objectives can be used as a conflict measure, where a more strongly negative correlation indicates a higher degree of conflict [54].

In reverse, this means that positively correlated objectives form synergies, which do not span a large space of trade-off options. Non-correlated objectives will generate random-based trade-off points or fuzzy data. From a system design perspective, this means that positively correlated objectives might not be real objectives, but rather an objective and a positively correlated constraint the extent of which is set by preferences. In order to detect such relations within the system, the correlation between objectives shall be monitored.

In the literature many well established correlation metrics can be found, first and foremost the frequently used PEARSON correlation. PEARSON can only detect linear relationships; the SPEARMAN metric r_s, however, is able to detect non-linear relations between two variable sets [55]. For the current system, it is unknown whether there are underlying relations at all, even though it is assumed for some objectives. In consequence, *Spearman* is chosen as an indicator for the degree of conflict, as it might detect more types of correlation between objectives than *Pearson*. The interpretation of

the values varies in the literature. In the current work the interpretations from [56] are used, which are listed in table 3.

TABLE 3: Interpretation of SPEARMAN coefficients according to [56]

$\lvert r_s \rvert$	interpretation of intensity
$\in [0.0, 0.3]$	**negligible** correlation
$\in (0.3, 0.5]$	**low** correlation
$\in (0.5, 0.7]$	**moderate** correlation
$\in (0.7, 0.9]$	**high** correlation
$\in (0.9, 1.0]$	**very high** correlation

When calculated pairwise for all objectives, the *Spearman* correlation values form a correlation matrix, which can be transformed into a heatmap plot indicating the conflict potential. The colored representation shown in figure 21 serves as a visual reference and allows spotting patterns or deviations across a series of experiments as well as different system configurations. With the aid of the generated heatmaps, a qualitative evaluation of conflict potential can be conducted.

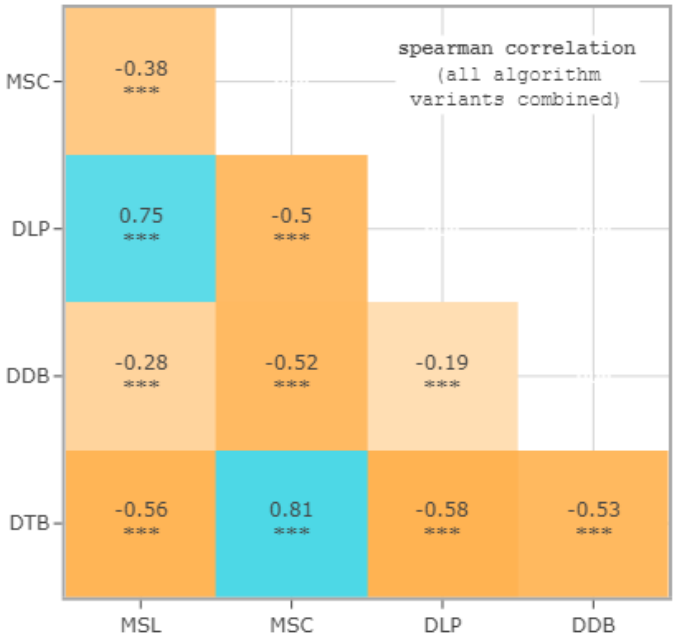

FIGURE 21: Example of correlation heatmap with color scale indicating conflicts and synergies

5.2.7 Normalized Hypervolume as comparable performance metric

In order to compare the performance of different algorithms, a performance metric has to be established. Otherwise it is not possible to say if e.g. a faster algorithm still delivers an adequate level of quality for its solutions. In the field of MOP, it is of special interest to retrieve the maximally diverse pareto front.

Every performance metric requires some kind of ground truth or reference to base the quantification on. In case of the IGD, this is a reference pareto front and for the hypervolume a dominated reference point is needed.

Even though the generated surrogate front could serve as an input for the IGD, this may have two drawbacks: This metric is biased towards the reference front and it is not strictly pareto-compliant [57]. Especially a lacking pareto compliance makes the IGD not always suitable for measuring overall performance [22].

It was mathematically proven by Fleischer [58] that for discrete MOPs, the maximum hypervolume indicates the true pareto front within the bounds set by the reference point. This pareto compliance motivates using the hypervolume for the given problem, as this problem is also of discrete nature. However, using a manually selected reference point is also a disadvantage: The calculated performance values are highly dependent on the selected position [59], which complicates a fair comparison. In the following approaches for solving this issue are discussed.

Quite frequently, the reference point is chosen to be the nadir point [50], although, different algorithms and even runs may generate different nadir points for each solution set. For a fair comparison of multiple algorithms, a reference point has to be defined which respects all solution sets. The required reference point is chosen to be based on the nadir point of the derived global surrogate front as suggested in [53]. Additionally, the solution set under investigation has to be scaled into the interval span by the ideal and nadir point of the global surrogate. Importantly, only solutions which dominate the reference point are allowed to be used in the hypervolume, which is ensured by the pymoo implementation of the hypervolume calculation [53].

While the solutions are scaled based on the exact ideal and nadir values, the exact reference point is set to be slightly worse than the nadir point ($1.1 \cdot z^{\mathrm{nad}}$). This method allows extreme points to also contribute to the hypervolume, which would not be the case when using the exact nadir point [53]. The procedure and benefits of offsetting the reference point is still subject to ongoing research and scientific discussions [60].

5.2.8 Moment of inertia as genome diversity measure

The previously introduced hypervolume metric is a measure of diversity for the pareto front, but this only takes the objective space into account. The described problem is a multi-modal one. Consequently, no conclusions about the genotypic diversity can be drawn from the objective space.

The concept of a shrinking living space shall emphasize exploration at an early stage of the algorithm by promoting genetic diversity. It is hoped that this method delivers at least the same performance as standard constraint-handling methods while providing solutions with a higher variation in decision space. First research results by AZEVEDO and ARAUJO focused on investigating the influence of genotypic diversity suggest that problems with disconnected pareto fronts benefit from enhancing diversity in the decision space [61]. In the best case, an even higher performance is thus also reached for the current problem in comparison to standard procedures.

AZEVEDO and ARAUJO use a diversity metric proposed by MORRISON and DE JONG which is inspired by the physical model of *moment of inertia*. Their model calculates the moment of inertia I created by individuals distributed in decision space with respect to the centroid C of the population P in decision space. The centroid vector components are calculated according to equation 26 and the moment of inertia is defined as in equation 27, which have been detailed in [62]. This metric is used in the presented work for evaluating the genotypic diversity of a solution set.

$$c_i = \frac{\sum_{j=1}^{j=P} x_{ij}}{P} \tag{26}$$

$$I = \sum_{i=1}^{i=N} \sum_{j=1}^{j=P} (x_{ij} - c_i)^2 \tag{27}$$

5.3 Evaluation from the perspective of system design

This part of the evaluation is focused on the analysis of the system itself, i.e. independent of the actual use case. The experimental series features a total of 48 different configurations. Within this section, notable effects shall only be illustrated by highlighting exemplary configuration instances, the full result set can be found in the appendix.

The system designed in this work features a combined approach of preference collection, detailed in section 3.3.2. During the conception phase, assumptions were made that different values of preset preferences may generate a varying conflict potential between objectives, as the preset preferences directly alter the objective function. In order to investigate the results concerning these assumptions, first scatter plot visualization of the surrogate fronts, generated from different preset preferences, have been examined with respect to shape and range of the front.

5.3.1 Algorithm configuration dependent effects

While initially planned for a comparison view between different systems, the surrogate front analysis of a single system already shows that there is a second source of variation, which is the algorithm configuration, i.e. the selection of operators used in the optimization run. In figure 22 the surrogate front analysis of system 8 across all algorithm is shown. The analysis consists of a scatter-plot matrix showing pairwise objectives and of a heatmap correlation plot of the surrogate solutions.

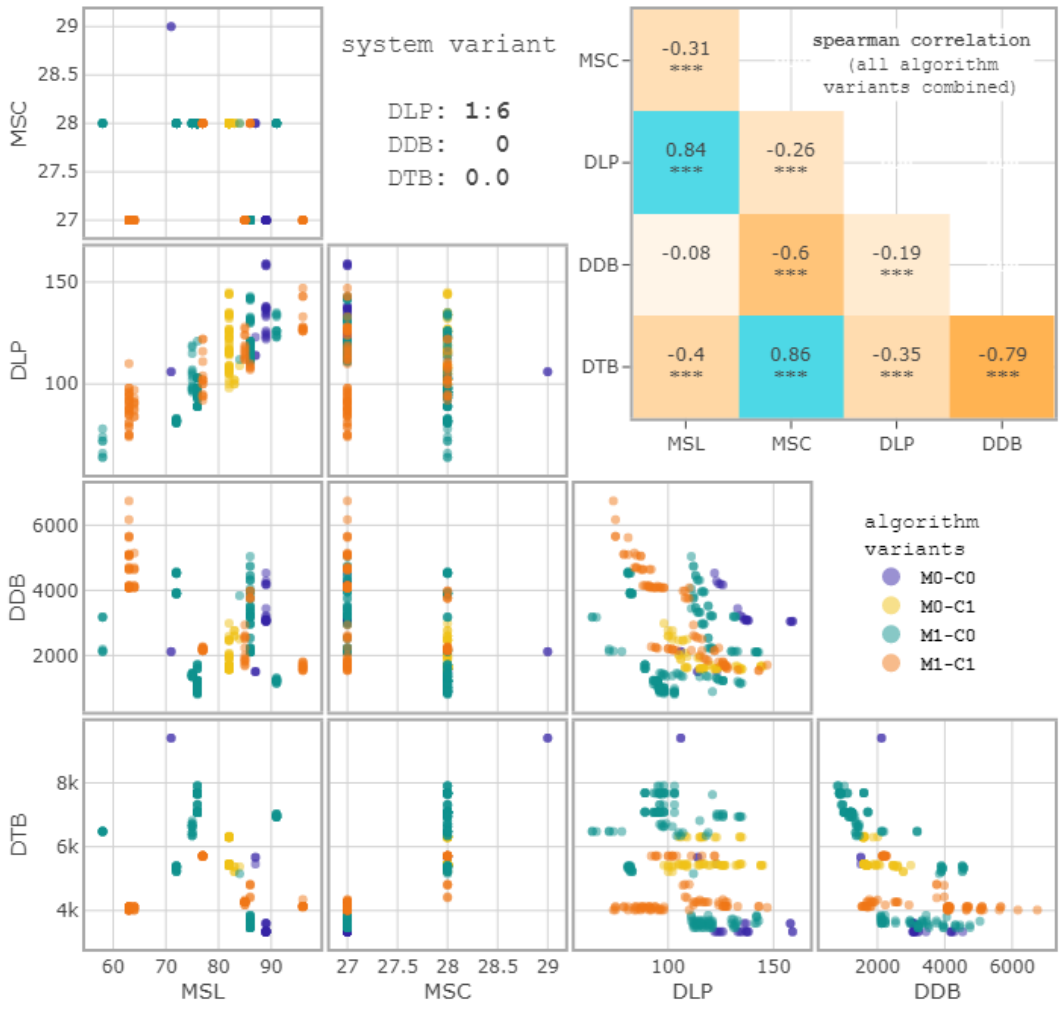

FIGURE 22: Surrogate fronts generated for system 8

The individual scatter-plot of an objective pair suggest that different algorithm configurations explore different parts of the objective space, clearly distinguishable by the color distribution. The initial expectation was to see rather similarly shaped fronts of varying magnitude, however the differing shapes of the surrogates for different algorithms indicate a potential variation in conflict potential.

In the following, the conflict potential variation within one system configuration across used algorithm is assessed firstly, before investigating variations suspectly caused by preset preferences. At this point, a more granular analysis of the conflict potential scoped to the different algorithm configurations is needed. Therefore, a correlation-heatmap plot is created for each configuration based on the local surrogate front, i.e. all non-dominated solutions from this algorithm configuration without domination-based filtering against other configurations. In figure 23 the resulting four conflict analysis are shown for system 8. The algorithm instance is indicated by the label, as well as coloring in correspondence to the introduced color mapping.

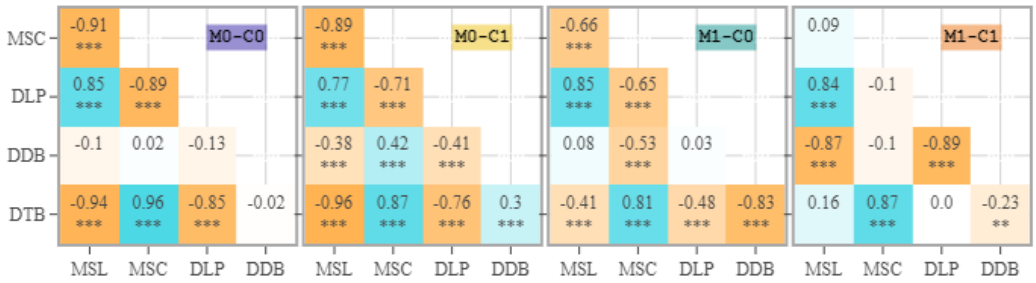

FIGURE 23: Conflict potential analysis for system 8 by algorithm

From figure 23 the following fundamental observations can be derived:
There are objective combinations which …

- … **do not** feature strong variation in their conflict potential across **all** algorithm configurations, e.g. $\{f_{\mathrm{MSC}}, f_{\mathrm{DTB}}\}$ ➔ type 1
- … **do** feature a strong correlation which softens gradually across **all** algorithm configurations, e.g. $\{f_{\mathrm{MSL}}, f_{\mathrm{DTB}}\}$ ➔ type 2
- … **do** feature a strong variation for a **single** algorithm configuration, e.g. $\{f_{\mathrm{DDB}}, f_{\mathrm{DTB}}\}$ ➔ type 3

These observations lead to the hypothesis that the conflict potential, which is measured by the presented metric based on aggregated run results, is not identical for each algorithm. Consequently, there has to be an influencing factor introduced by the algorithm that distorts the nature of conflict potential contained within the system itself. To cover this aspect, it shall be distinguished between the *real conflict potential*, i.e. a potential which is assumed to be an inherent part of the system itself, and the *observed conflict potential*, i.e. a potential that is assumed to be either highlighted or shadowed by the applied algorithm.

In addition to the primary observation, an underlying pattern in the change of algorithm configuration may be described as following: For type 2 objective combinations, the correlation variation seemingly corresponds to the introduction of operators focused on explorative behavior (*sprint-swapping mutation*, *shrinking living space*). However, no clear functional relation can be derived which is valid for all objectives. E.g. upon introduction of explorative operators, the correlation of $\{f_{\mathrm{MSC}}, f_{\mathrm{MSL}}\}$ decreases, while it increases for $\{f_{\mathrm{DDB}}, f_{\mathrm{DLP}}\}$. Therefore, no general rule can be formulated based on the currently known data.
With respect to the system under investigation, three hypotheses for potential interpretations are made:

1. A conflict potential which persist across all configurations (type 1) is likely to be a *real conflict potential*.
2. A conflict potential that gradually changes (type 2) is likely to be an *observed conflict potential*, which is dependent on general parameters, e.g. the degree of explorative behavior.
3. A conflict potential that has a peak variation for only one configuration (type 3) is likely to be an *observed conflict potential*, which is sensitive to a single operator or combination of operators.

This means, that not only the general target (e.g. explorative behavior), but also the interdependencies of different operators may be an influencing factor.
These empirical observations can be found in multiple systems from the experimental series which backs the previous considerations. However, the number of algorithm configurations is far too little to investigate the operator influence in-depth. Additionally, the different configuration instances are too diverse for a more precise quantitative analysis. Consideration of a possible experimental setup to further investigate the suspected influences will be addressed in future work.

5.3.2 Preset preference dependent effects

Aside from the variation introduced by the algorithm configuration, there is also a potential influence caused by the preset preferences. Even during the system draft some of these preferences were expected to have a varying conflict potential, most notably the $\{f_{\mathrm{MSC}}, f_{\mathrm{DTB}}\}$ combination, i.e. lowering the preferred fill grade may increase the number of sprints used and vice versa. In the following, the system is examined concerning such effects.

Three main effects can be observed in the experimental series, which are illustrated by using the heatmap comparison view in figure 24 and the following objective pairs:

- ① denotes $\{f_{\mathrm{DLP}}, f_{\mathrm{MSL}}\}$
- ② denotes $\{f_{\mathrm{DTP}}, f_{\mathrm{MSC}}\}$
- ③ denotes $\{f_{\mathrm{DTB}}, f_{\mathrm{MSL}}\}$

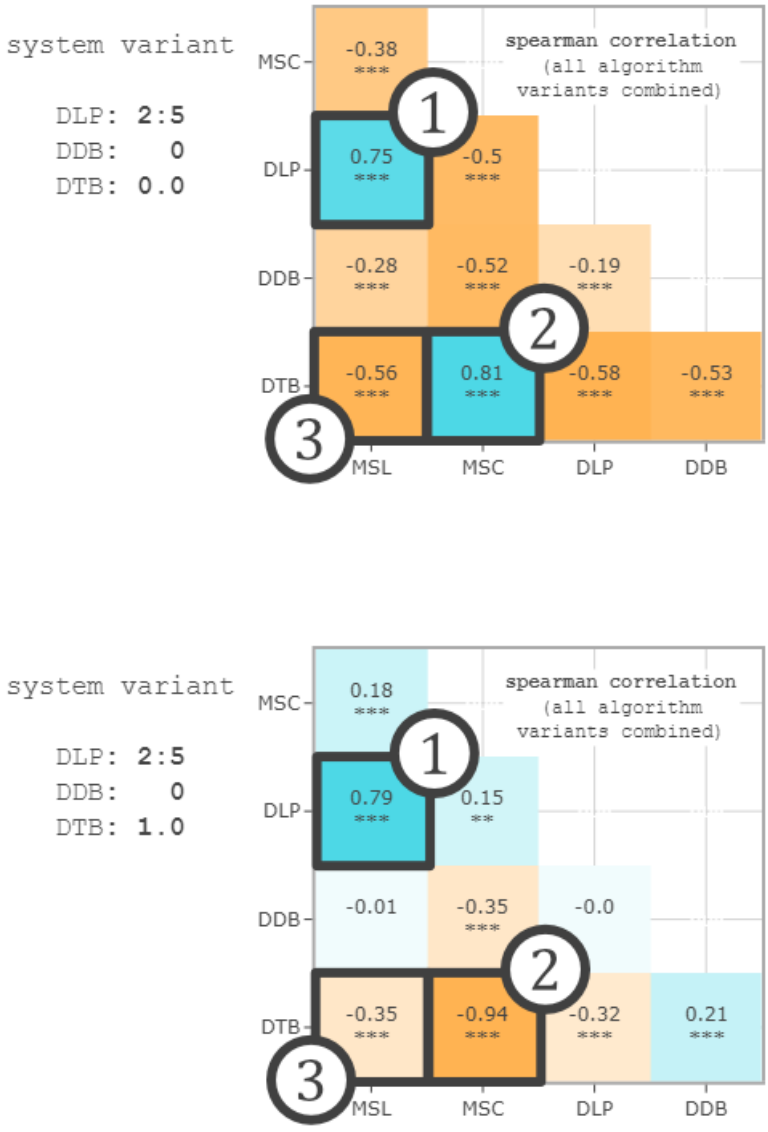

FIGURE 24: Comparison of variation in conflict potential caused by preset preference

The analysis from the previous section showed that the correlation analysis is a tool that has to be used with caution. Nevertheless, it is a valuable tool for gaining insights. In order to prevent wrong interpretations, only effects that persist across similar systems or show a regular pattern corresponding to configuration changes are described in the following.

The first effect describes correlations that do not change across configuration changes, e.g. ① ➔ objective combination $\{f_{\text{DLP}}, f_{\text{MSL}}\}$. This synergy effect means that as soon as one of both objective values decreases, the other does so. For this objective pair, similar correlation values have been observed in 47 out of 48 systems, which clearly indicates that the strong synergy is stable across all configurations and persists against varying presets as well as algorithms.

The correlation analysis therefore helps to form an understanding of the system behavior, especially concerning rather fuzzy relationships. As illustrated by figure 25, the relation between the two objectives based on last sprint index and editing distance might have a fluctuating relationship. The experimental results indicate that for the given data, no conflict exists, despite being possible at least for on a theoretical level. Such analysis are rather difficult to derive from the pure function definitions.

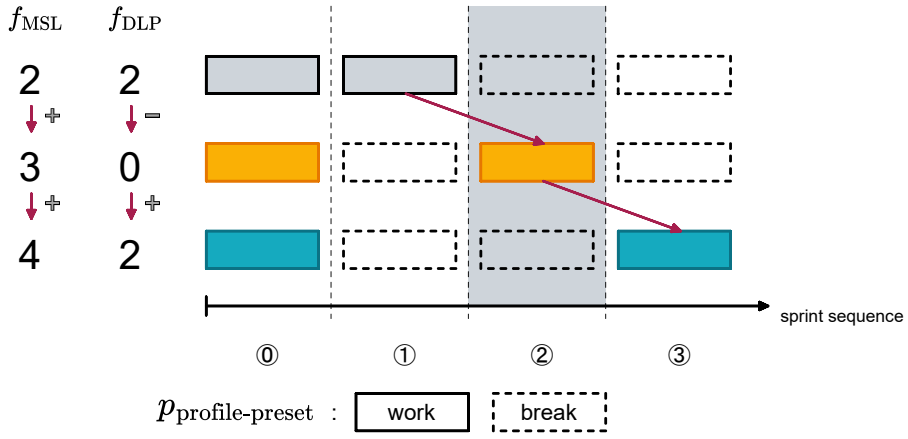

FIGURE 25: Potential relationship directions of the DLP and the MSL objective

Another example for such verification can be found with respect to the DDB objective. Based on the function definition alone, a change in preset preference may appear like a straight shift so that it is simply optimized towards another target value without changing the conflict potential. Despite this impression, the experimental results suggest that the system reacts sensitive to a preset preference concerning that objective. The reason for this may be the chosen encoding which decreases the number of "acceptable" sprints for low $p_{\text{deadline-buffer}}$ values, as indicated in figure 26.

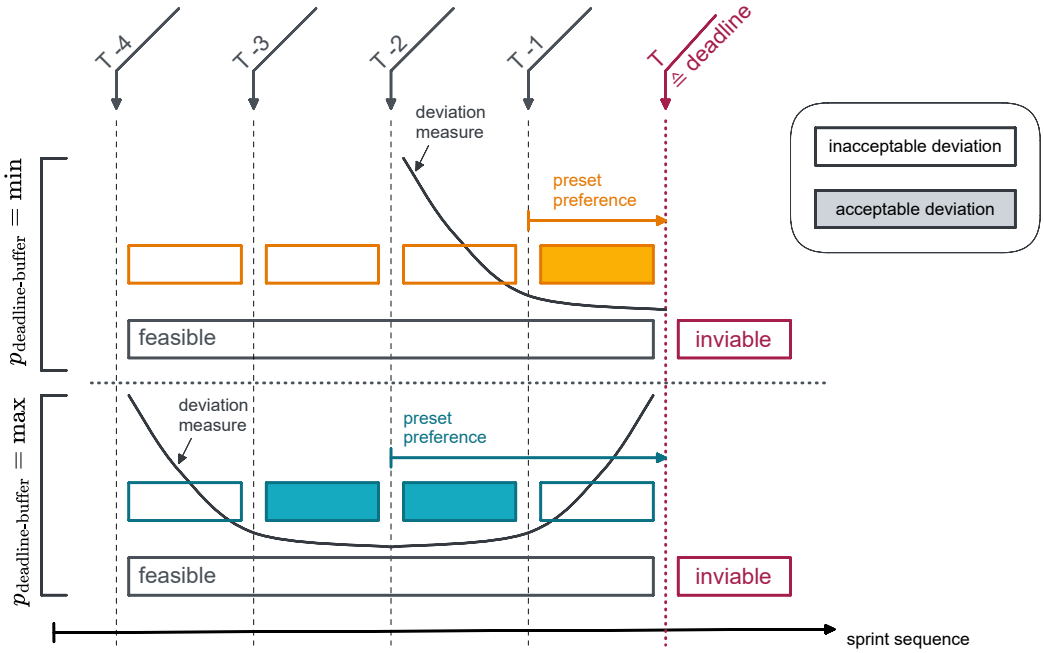

FIGURE 26: Unbalanced effect of preset preference due to system bias

However, there are also objectives pairs with a predictable correlation that also corresponds to intuitive assumptions. For example, DTB and MSC form a very intuitive conflicting relation. As suspected during the system draft, a $p_{\text{target-buffer}} = 0$ will make the *deviation from target buffer* objective almost equal to the *minimum sprint count* objective, i.e. causing a strong synergy effect. If the preference is reversed, the effect is reversed as well, so the preset preference influence is clearly predictable and a direct influence.

The third effect, which can be observed relates to the $\{f_{\text{DTB}}, f_{\text{MSL}}\}$ combination. Intuitively, a higher $v_{\text{fill-grade}}$ would result in a shorter schedule, as fewer sprints are consumed. The experimental results clearly indicate the opposite: higher fill grades conflict with shorter schedules. This observation is counterintuitive, but a hypothetical explanation can be formed: While the intuitive assumption reflects the static system state, the experimental results reflect a state which is based on a dynamic history. As indicated in figure 27, moving a sprint from the tail section of a schedule is more difficult when there is less capacity available in other sprints.

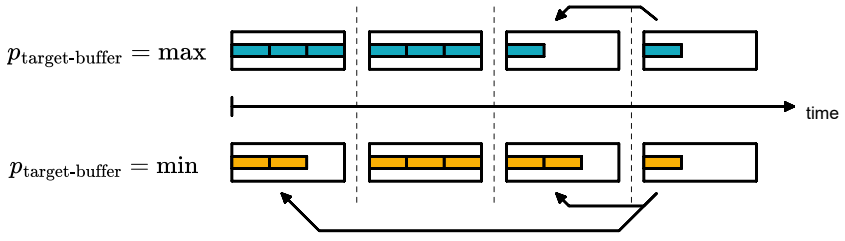

FIGURE 27: Dynamic conflict potential influenced by preset preference

5.3.3 Basic sensitivity analysis of custom operators

As a last step in the evaluation of the implementation, a basic sensitivity analysis shall be conducted. As a full-scale analysis would require more granularly changing configurations. For the first test of this newly created system model, only extreme values in combination with standard hyperparameter values were run. Consequently, only general trends can be derived from the results and no granular results.

For a structured comparison of different preset preference values in combination with different algorithm configurations, a confusion matrix style representation was created which is shown in figure 28.

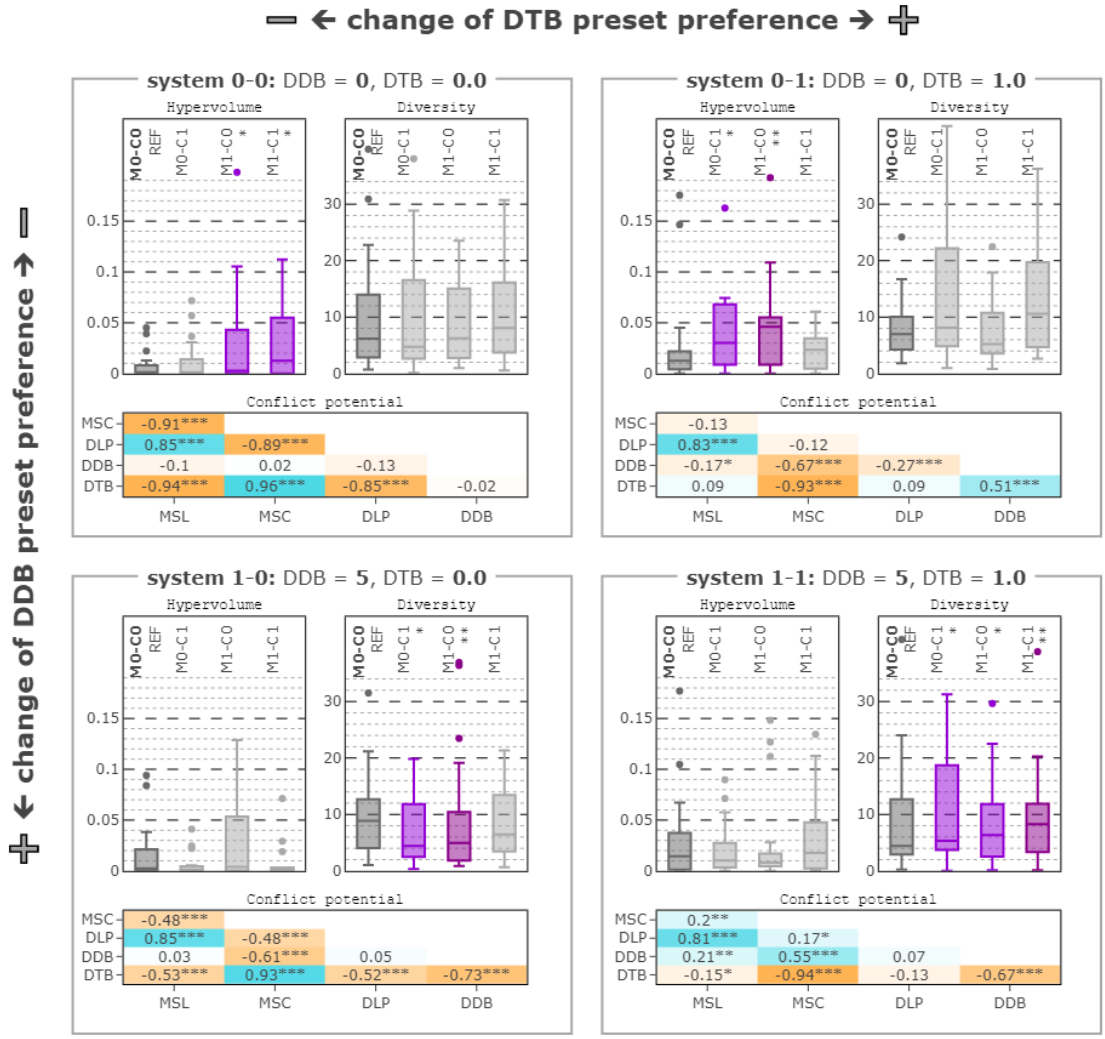

FIGURE 28: Hypervolume, diversity and conflict potential for systems with a preset of 1:6 (colored in case of significance)

Each of the four field corresponds to one combination of the DTB and DLP preset preference values. Based on the investigations from the previous sections, it is assumed that higher preference values for these objectives results in a relaxation of the constraint level of the system, which also correspond to the overall impression of the correlation heatmaps for these systems. Figure 28 presents the DLP configuration with the highest number of algorithm configurations that obtained significant results during the experimental series, the remaining two result sets can be viewed in the appendix.

In terms of hypervolume, the custom operators performed either significantly better or not significantly worse than the default implementation in 11 out of 12 each system instance. Judging from the number of significant results obtained, exploration-focused custom operators appear to be beneficial for the optimization in higher constrained systems.

The underlying effects for the obtained values of the measures are not fully understood, yet, as in some systems it is significantly worse than the vanilla implementation. However, in other system instances, the diversity value is significantly better without a clear pattern which corresponds to the variation in preset preference values. A first step for a comprehensive understanding may be testing alternative diversity metrics to check whether these results are reproducible.

Concerning the implementation approach of a shrinking living space and a proximity-oriented mutation, the trends in the results look promising. However, based on the results of one system which is not a general benchmark system, no serious statements can be made concerning the general efficiency and applicability of this design component.

5.4 Evaluation from the perspective of practical application

Minimum Schedule Length – The experiment results suggest that a minimization trend is generate. For some systems, this trend is far from being at a feasible level for a real-world use case. Especially, in the tail section of a schedule large gaps occur that are created by unused sprints which can occupied without issues, at least in theory. There are two main hypotheses for this phenotypic occurrence:
Firstly, the introduced *sprint-swapping* mutation might be under powered which could be easily tested by further test run with varying mutation rates.
Secondly, the objective formulation is potentially not well suited for the intended result. The chosen formulation bases the objective value only on the index of the last sprint used, which could be a potential design flaw, as this index does not consider the schedule as a whole. An alternative formulation for future research might be considering some kind of "balancing"-index, that calculates the distribution of used sprints and rewards high numbers of used sprints in the head section while penalizing high numbers of used sprints in the tail section of a schedule.

Minimum Sprint Count – The experiment results show that for each system configuration at least one algorithm configuration was able to reach a lower bound of sprints used, i.e. 27 sprints. The optimal number of sprints specified by the benchmark is 25 [46]. With respect to the numerous other objectives, the experimental result is regarded as a success, which means that the implementation provides a reasonable solution for a real-world use case. Importantly, the algorithm configurations which included custom operators reached that minimum more often than the vanilla version.

Deviation from Load Profile – Phenotypes obtained during the experimental testing show a tendency of groupings according to a pattern specified in the *preset preferences* which suggests that using the LEVENSHTEIN distance as a metric is a working approach for pattern generation. However, for the use case of the specified *preset preferences* it is considered only partly applicable, since marginal variation in the specified preset value do not cause an exact alignment. On the one hand this might be the result of interactions with the mutation operators, on the other hand the quantification of deviation values may not be granular enough. A potential approach for future experiments would be decorating the LEVENSHTEIN distance with a scaling function.

Deviation from Target Buffer – Judging from the phenotypes of the results, the optimization performs well at arranging the schedule according to the chosen *preset preference*. A potential a source of misinterpretation are strong deviation values for high-buffer preferences. This is supposedly due to the data set used, since it contains activities with a duration equal to the capacity of a sprint, so that a schedule with strong deviations is the only option. While this could be mistakenly interpreted as an algorithm flaw, it is regarded as an anomaly caused by the data-set.

Deviation from Deadline Buffer – The experiment results indicate that a clear minimization trend of the objective is possible with the chosen formalization and NDF feasibility is ensured. Despite being a powerful constraint handling technique, the chosen encoding may become a bottleneck of scaling with respect to real-world data. The adjustments to the data set created a very controlled environment for experiments. Introducing a higher number of deadlines and relaxing some strict requirements to the data set, may constrain a system quickly to an unsolvable state. However, the decision of applying such restrictive implementation is highly use-case dependent. For a personal scheduling system this may be unusable, while it may be an indispensable error-prevention measure in e.g., scheduling automated actions in critical infrastructure.

Viable at any time requirement – The implementation of a true *viable at any time* system has not been tested in the experimental evaluation, since developing an adequate repair operator is considered beyond the scope of this work. The general classification, however, is regarded as a valuable design aid, especially for real-time systems.

Incorporating real-world attributes – The current solution is built on an abstract level i.e., representations of real-world aspects like calendar systems are not covered. The system conception in general allows to incorporate these features with just minor effort. A calendar can be represented by introducing another mapping function of sprint indices to dates (which is also compatible with the chosen implementation of deadlines). Fixed calendar events may be added by altering the capacity of sprints which coincide with the given date. However, advanced real-world attributes like sequence information of dependent activities can not be depicted in the current solution without major changes.

6. Concluding summary

In this work, the entire process of developing an optimization for a real-world problem was gone through. Starting with the motivation of why there is demand for an individualized scheduling solution, the a real-world use case was examined. Existing scheduling procedures were considered and individual system elements derived from them. Based on these components, standard problems in computer science were considered with regard to their suitability as a basis for an own problem description. A feasible scope for a system model was defined based on an assessment of the computational complexity of the standard problems investigated and the actual requirements of the use case. Further developments were then based on the existing standard problem of bin packing.

In order to derive an abstract system model from the real application case, an observation technique was introduced: Starting from the user preferences, objectives were defined and then constraints were defined based on these. The resulting theoretical descriptions of the system elements were then transformed into an abstract mathematical system model featuring all aspects of a multi-objective problem. As part of this formalization, a non-standard method of preference collection was introduced.

In this approach, a priori and a posteriori methods are combined, so that the user can already specify a basic direction before optimization begins. In this way, the algorithm is automatically guided into areas of the search space that correspond to the user's basic attitude. After the end of the optimization, the user can choose from a diverse set of solutions which he wants to have according to his trade-off preference.

Following the formalization, possible options of the implementation were considered. Based on the similarity with BPP concerning the computational complexity and due to the non-applicability of existing heuristics, it was decided that metaheuristics, in particular genetic algorithms, are applicable to the custom problem.

In preparation to implementation, the system was analyzed in terms of infeasibility and a new classification was proposed for indicating the requirements concerning constraint handling.

NSGA-II as state-of-the-art algorithm was chosen as basis for the implementation. In the following, algorithm components for this system were designed and implemented for optimization with the help of the standard library pymoo. Special attention was paid to compliance with the defined system specifications in terms of infeasibility. Notably, an encoding was chosen that already respects most of the constraints and does not violate them.

For the actual optimization, in addition to the standard operator (uniform cross-over), a separate mutation operator was developed with the aim of better proximity. For this purpose, a second representation level was also introduced in the encoding.

Since the constraint of a minimum fill grade makes the system even more difficult to solve, a separate solution for constraint handling was also introduced: the concept of the shrinking space. In this method, the exploration is supported by tolerating actually infeasible solutions at the beginning

of the optimization, while at the end the constraint is enforced to the full extent, so that infeasible solutions die out.

To test the implementations, an existing benchmark dataset was modified and an experimental setup was designed. Particular attention was focused on the effects on conflict potentials when the algorithm configuration or the preset preferences and therefore the system configuration were changed. To investigate these aspects, the hypervolume, an inertia moment-based diversity metric and a correlation analysis were used. Since different algorithm configurations are sometimes difficult to compare, the concept of a surrogate front was implemented.

In the course of the experiments, it became apparent that there seems to be a relation between algorithm configuration and conflict potential. Consequently, the distinction between real conflict potential and observed conflict potential was introduced.

In addition to this finding, it was also shown that, as expected, the preset preferences also have an influence on the conflict potential. However, this influence does not always behave intuitively. In this context, a further distinction between static and dynamic conflict potential was proposed.

Finally, it was looked at the general performance of the custom operators, some of which were significantly better than the vanilla variant in strongly constrained systems. Since these operators had a rather explorative character, it is assumed that this has a positive effect on the solution of complicated systems. In most of the less constraint systems, there was no significant reduction in performance compared to the standard case.

7. Future work

The presented work had a relatively high explorative approach, i.e. multiple untested concepts have been combined. In development, each diverging phase has to be followed by a converging phase, which streamlines discoveries. Consequently, the different approaches have to be tested on known standard problems in isolation.

An important aspect of designing an optimization with evolutionary algorithms is proper parameter tuning. In future research, the investigation of the influence of parameter selection paves the way for closer investigation of the effects related to varying conflict potential of objectives. Gradually changing parameter setting would allow studying the interrelation between an increasing diversity and the correlation of objectives.

During such future research efforts, potentially another diversity metric for analyzing the diversity in decision space may be required as the current experiments were relatively vague concerning the influence of the diversity metric.

The concept of a shrinking living space provide promising results, but the test problem is not representative. In future experiments, this concept may be applied to constraint benchmark problems with a known pareto front, so that decisive evaluation of the method can be done. Additionally, the possible short-coming of a high number of function evaluation for the retrospective re-evaluation of all individuals could be addressed by moving from a continuous shrinkage to a stepwise one. Developing strategies for this is subject to future research.

With respect to the actual use case, refinement of the objective definitions and tests with real data form an interesting perspective for further research, even though the system is at a low technology readiness level.

Appendix

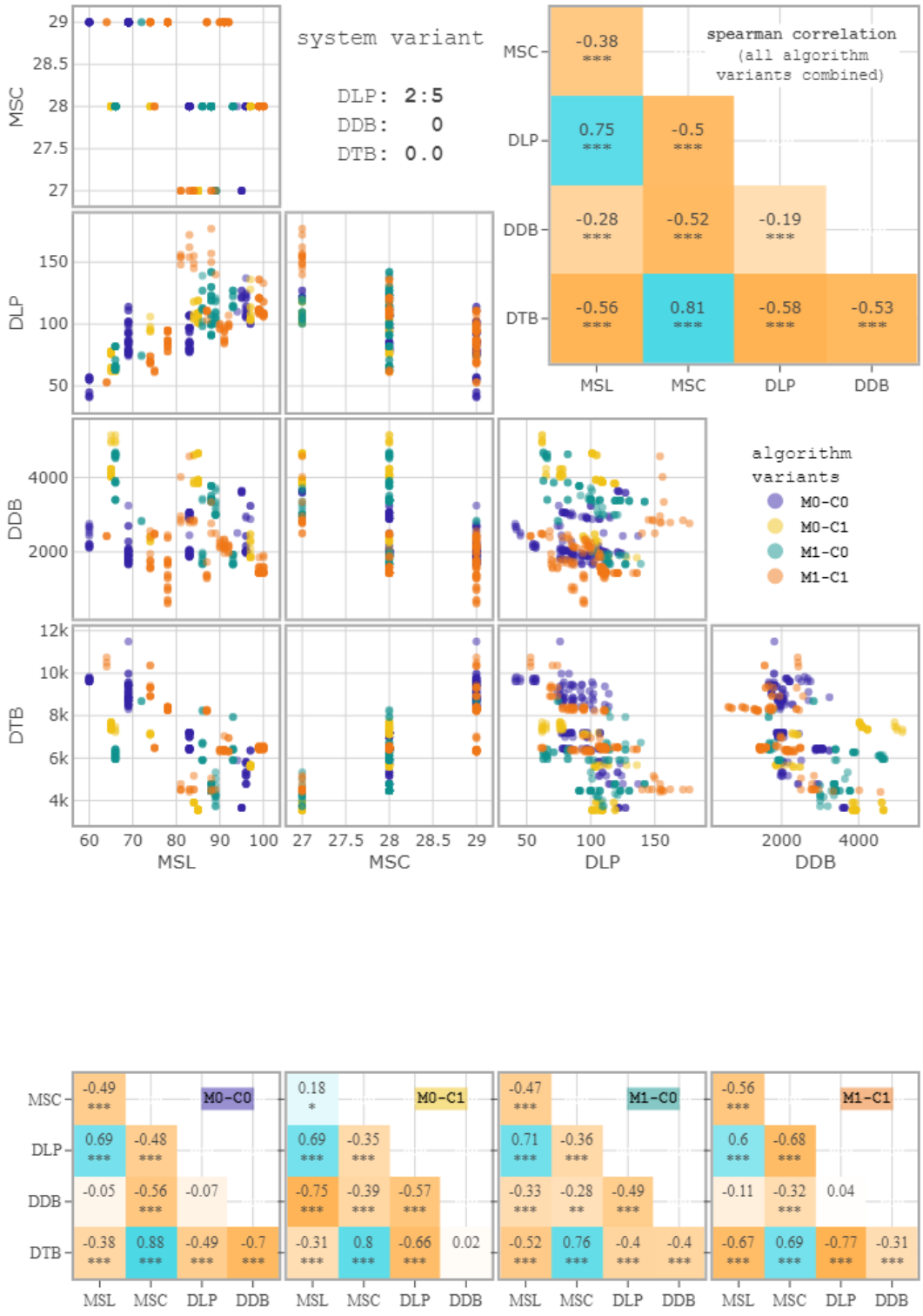

FIGURE A-00: Result set of system configuration 0

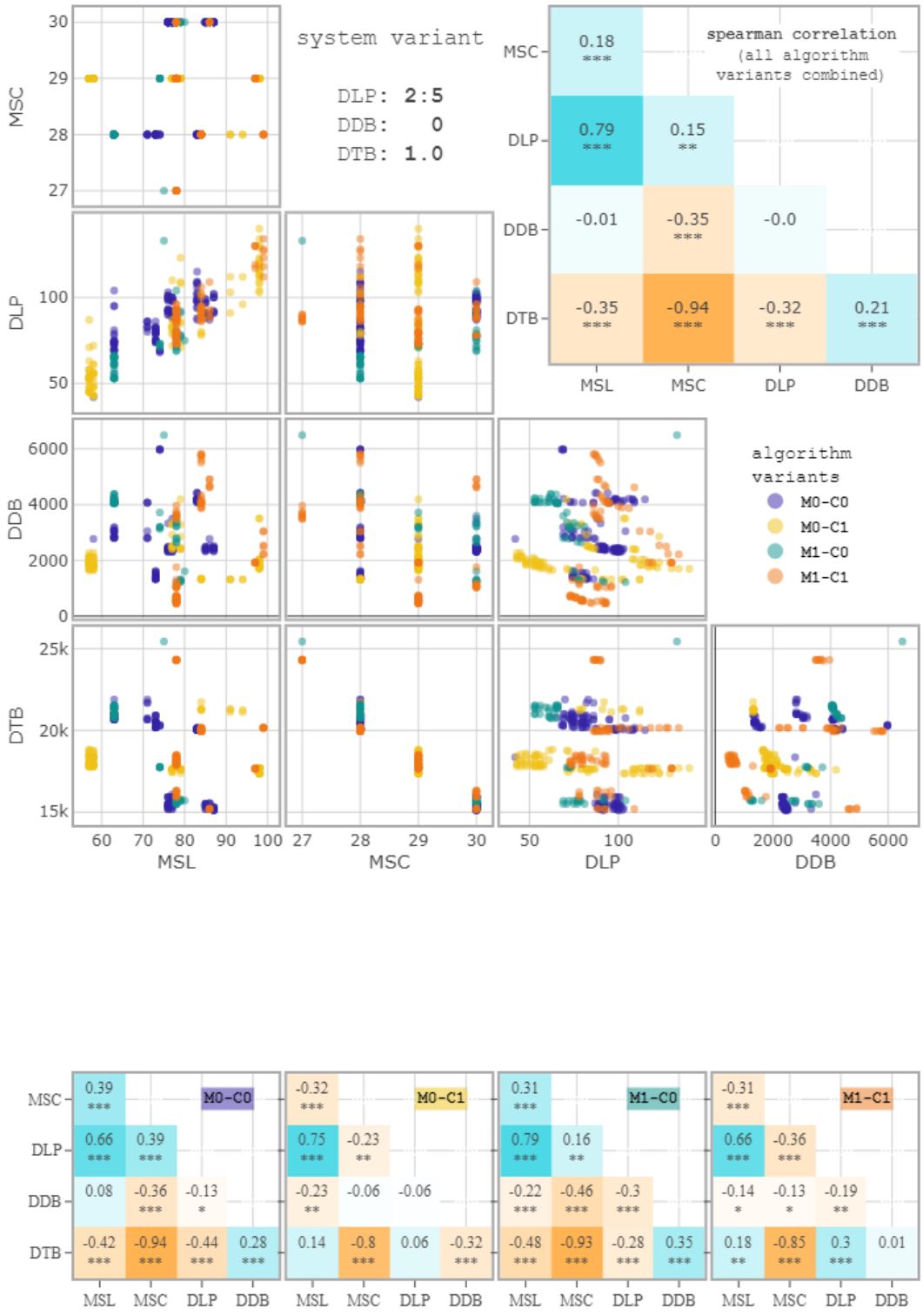

FIGURE A-01: Result set of system configuration 1

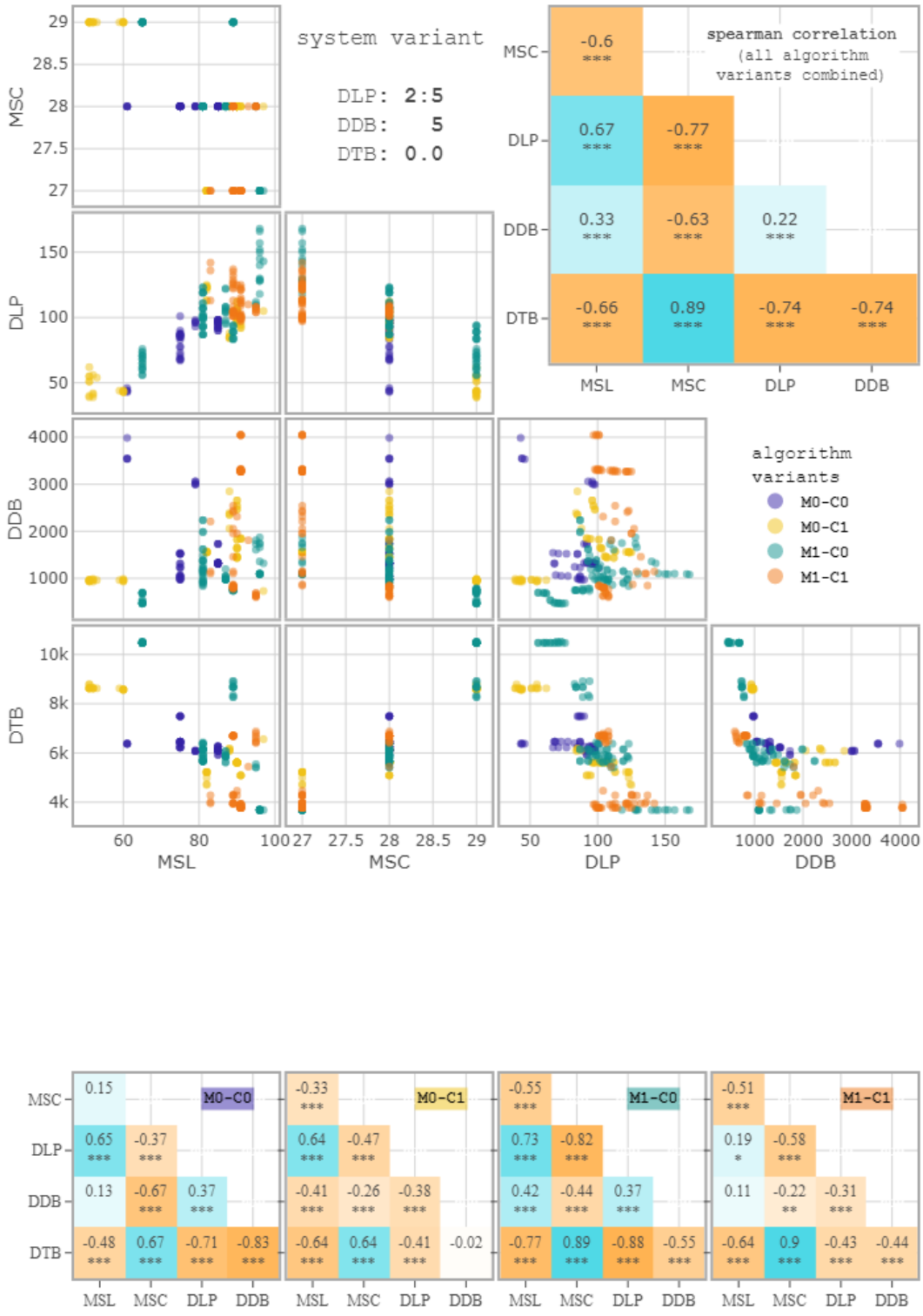

FIGURE A-02: Result set of system configuration 2

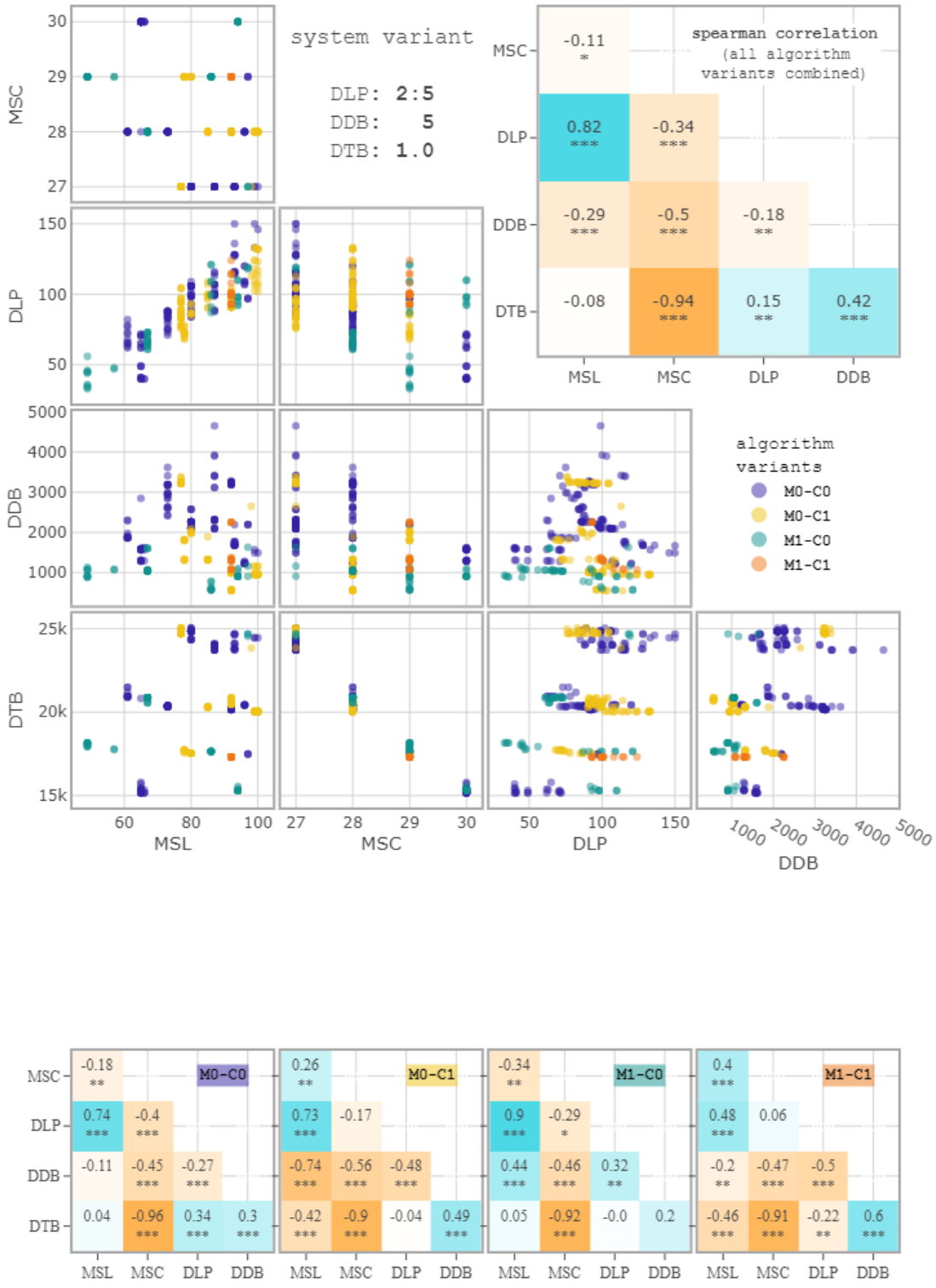

FIGURE A-03: Result set of system configuration 3

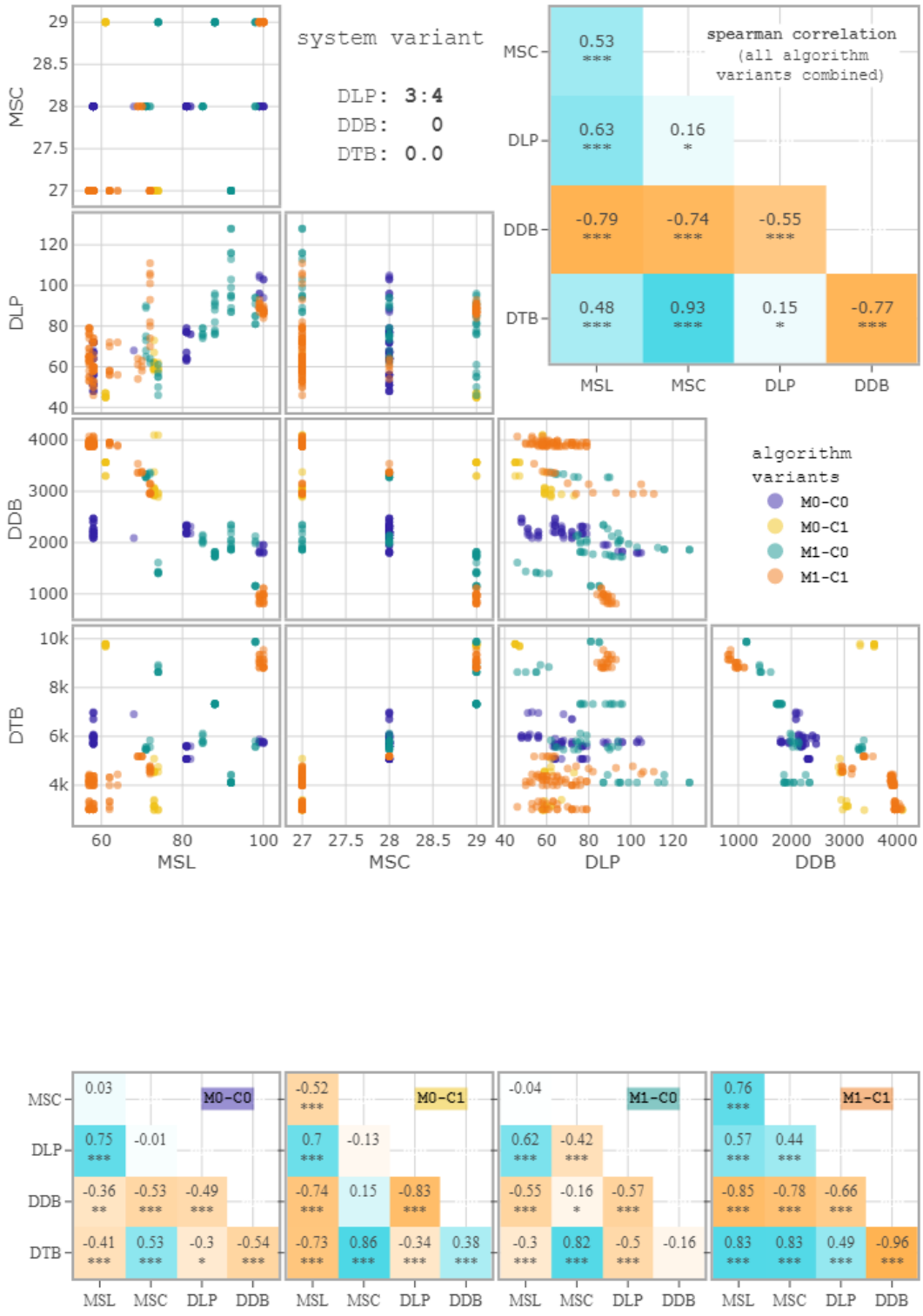

FIGURE A-04: Result set of system configuration 4

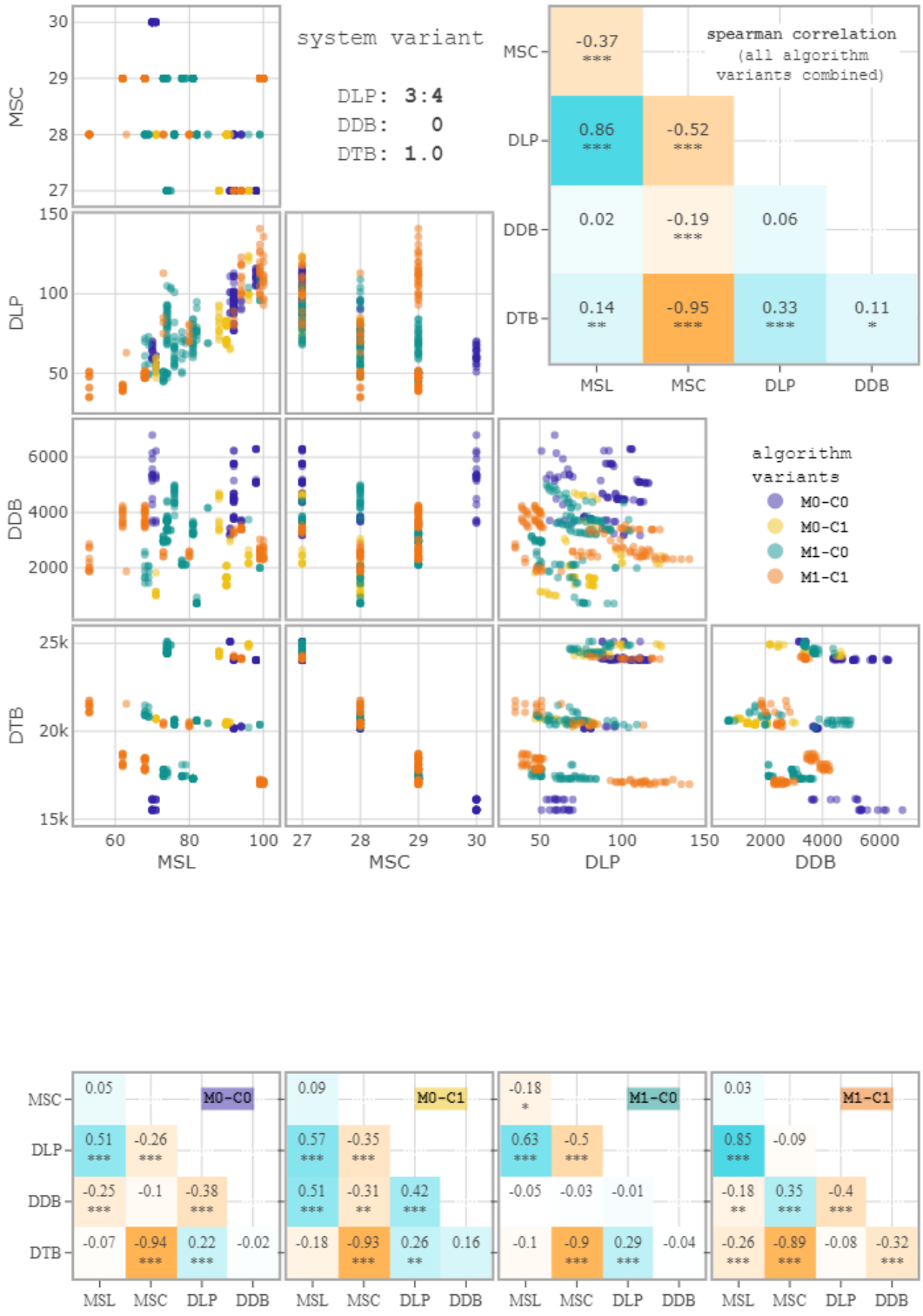

FIGURE A-05: Result set of system configuration 5

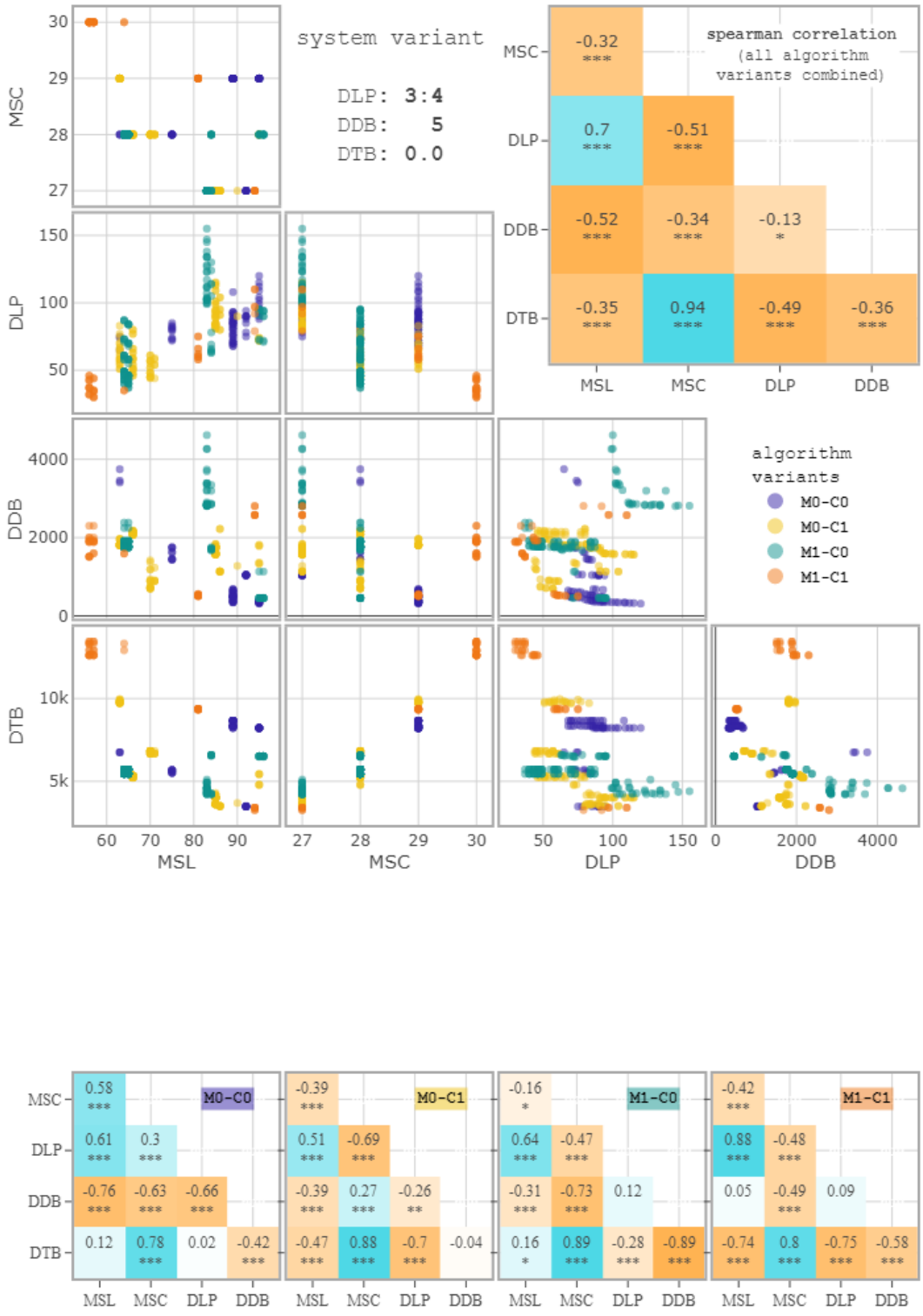

FIGURE A-06: Result set of system configuration 6

FIGURE A-07: Result set of system configuration 7

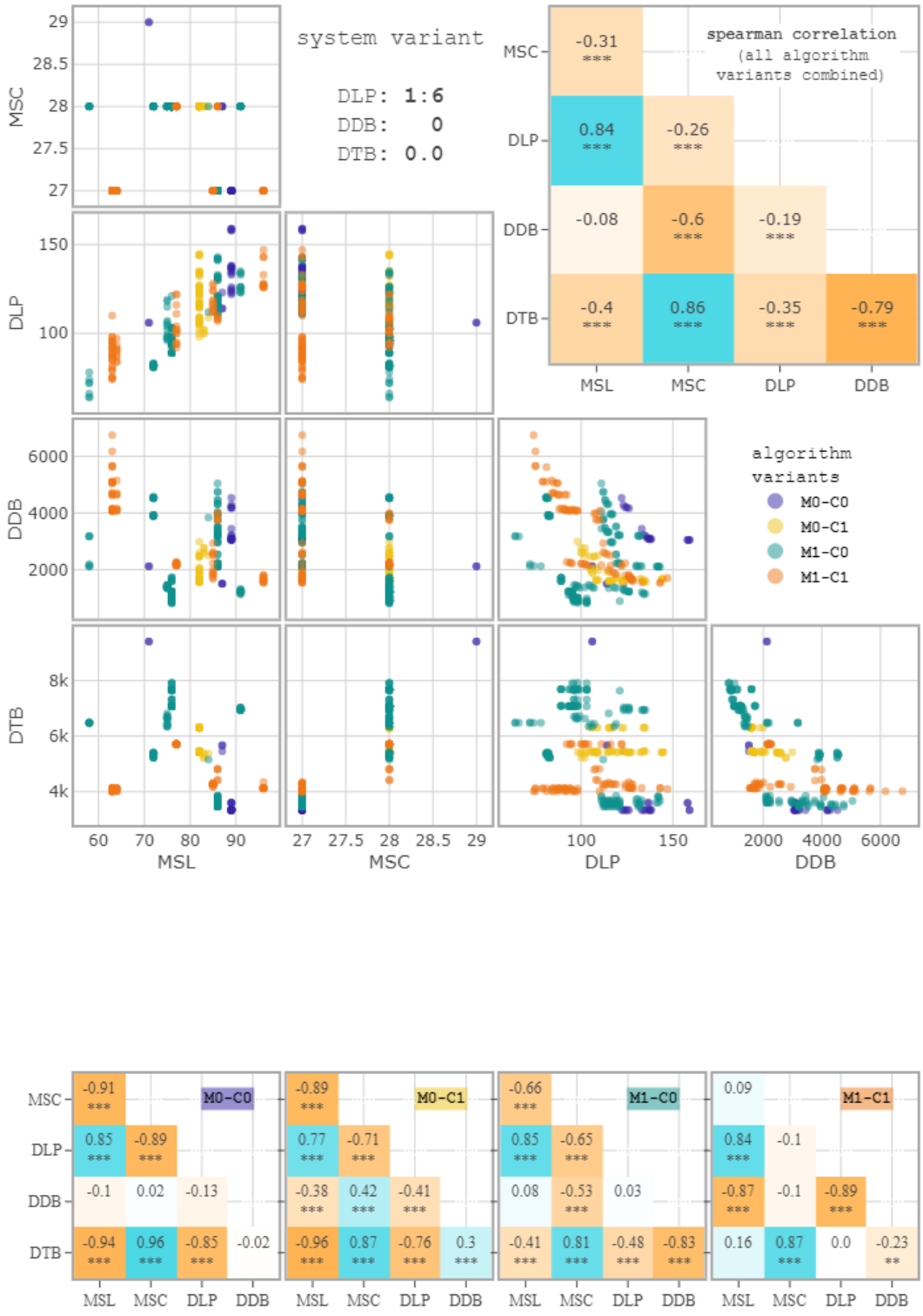

FIGURE A-08: Result set of system configuration 8

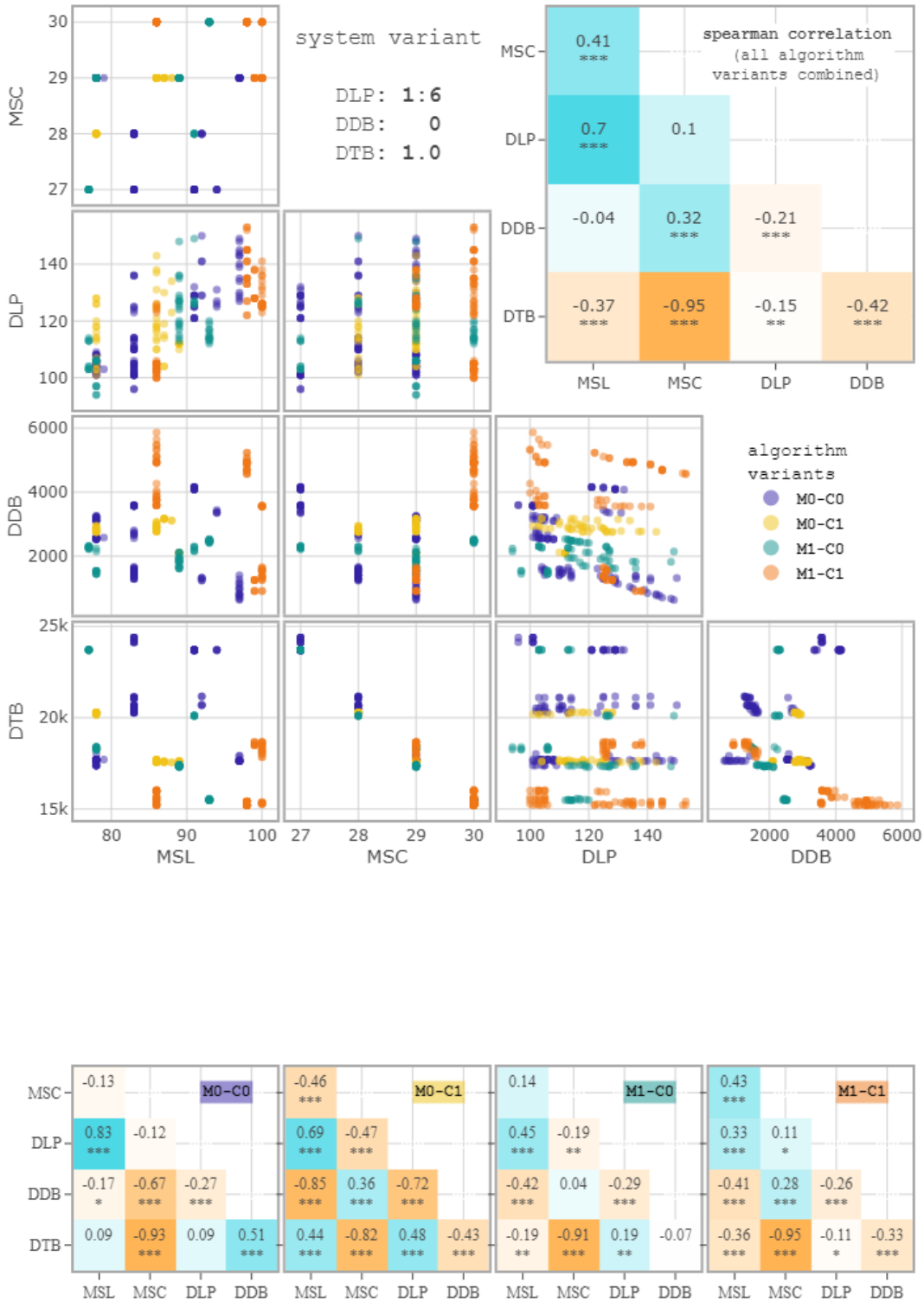

FIGURE A-09: Result set of system configuration 9

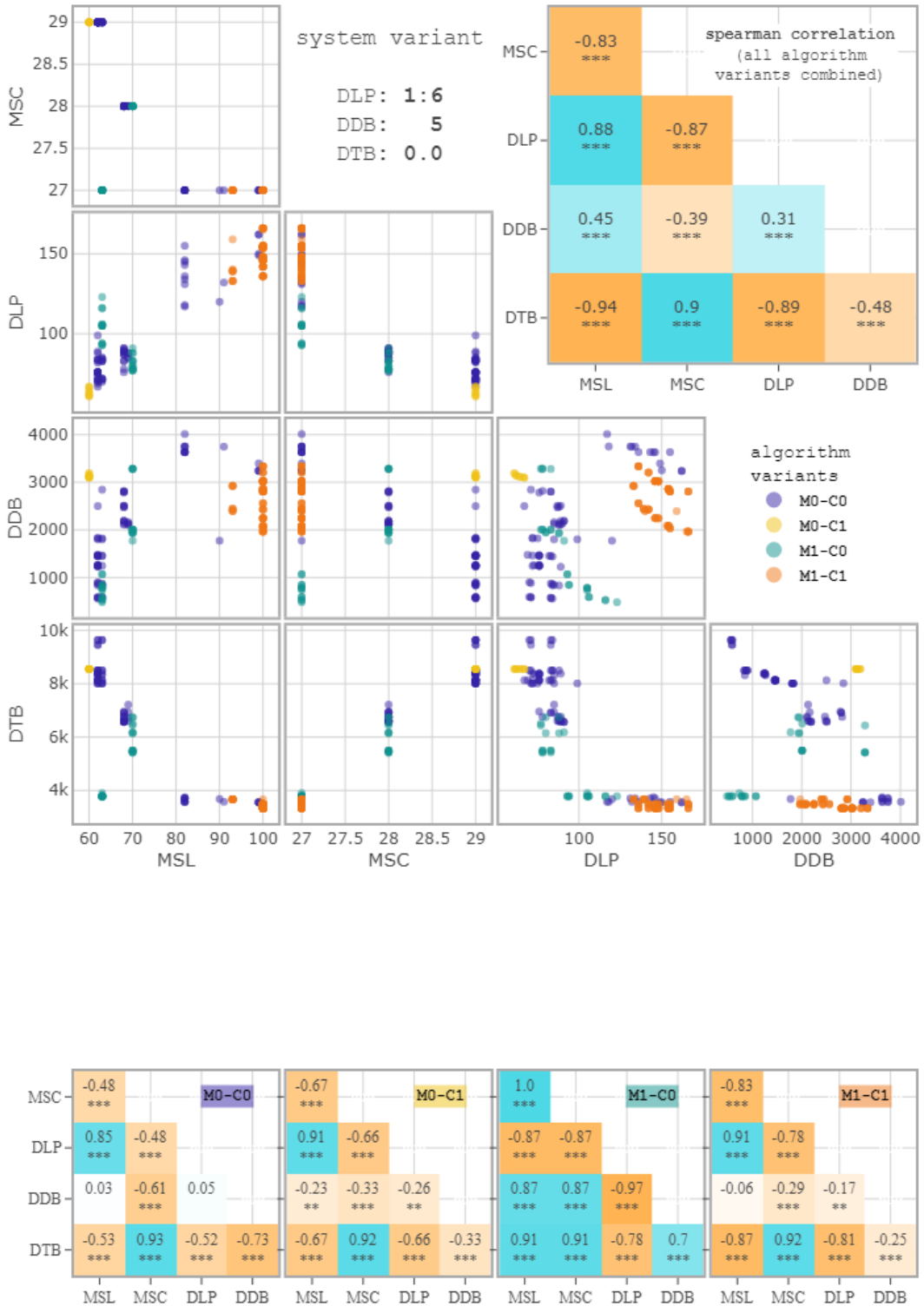

FIGURE A-10: Result set of system configuration 10

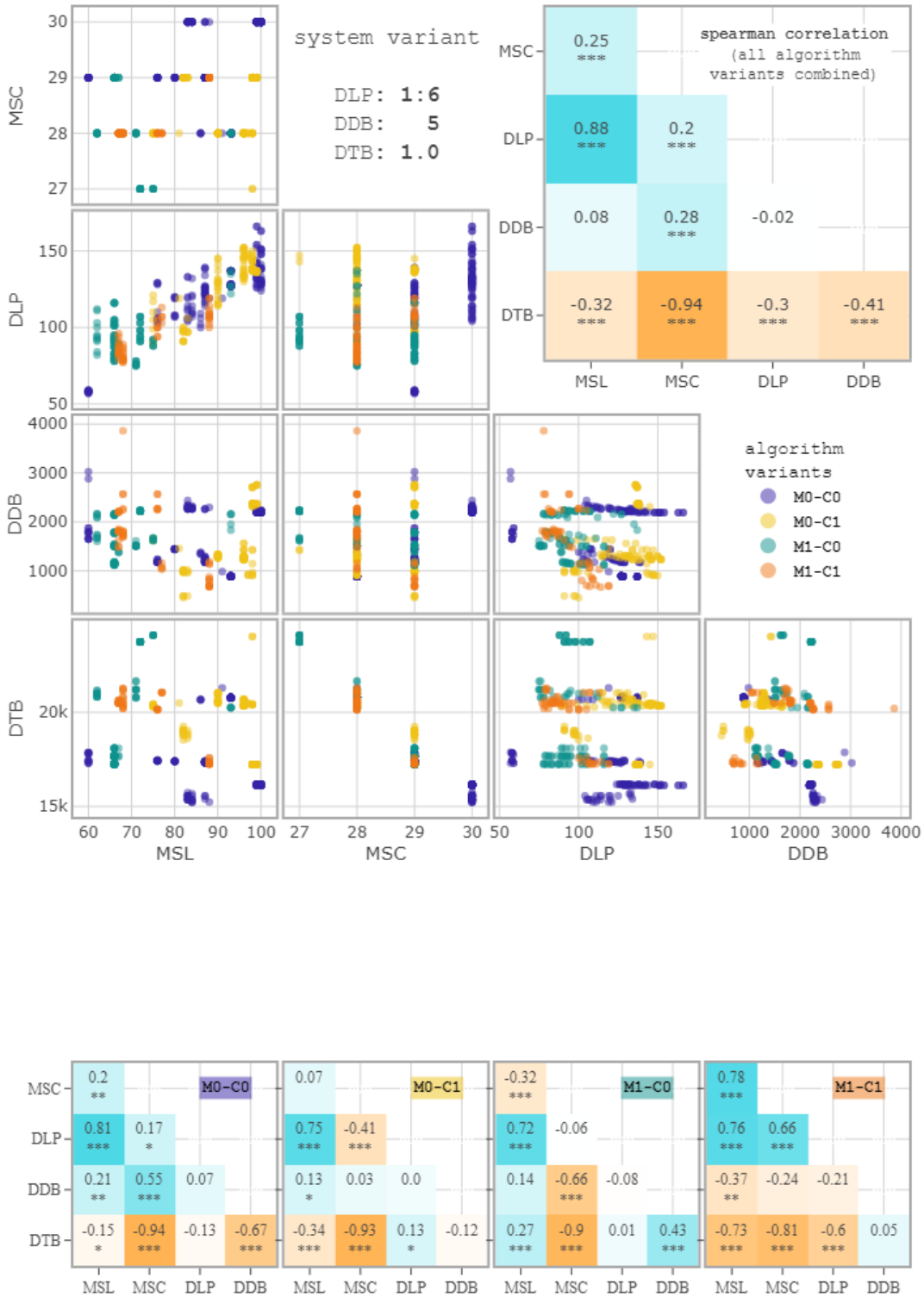

FIGURE A-11: Result set of system configuration 11

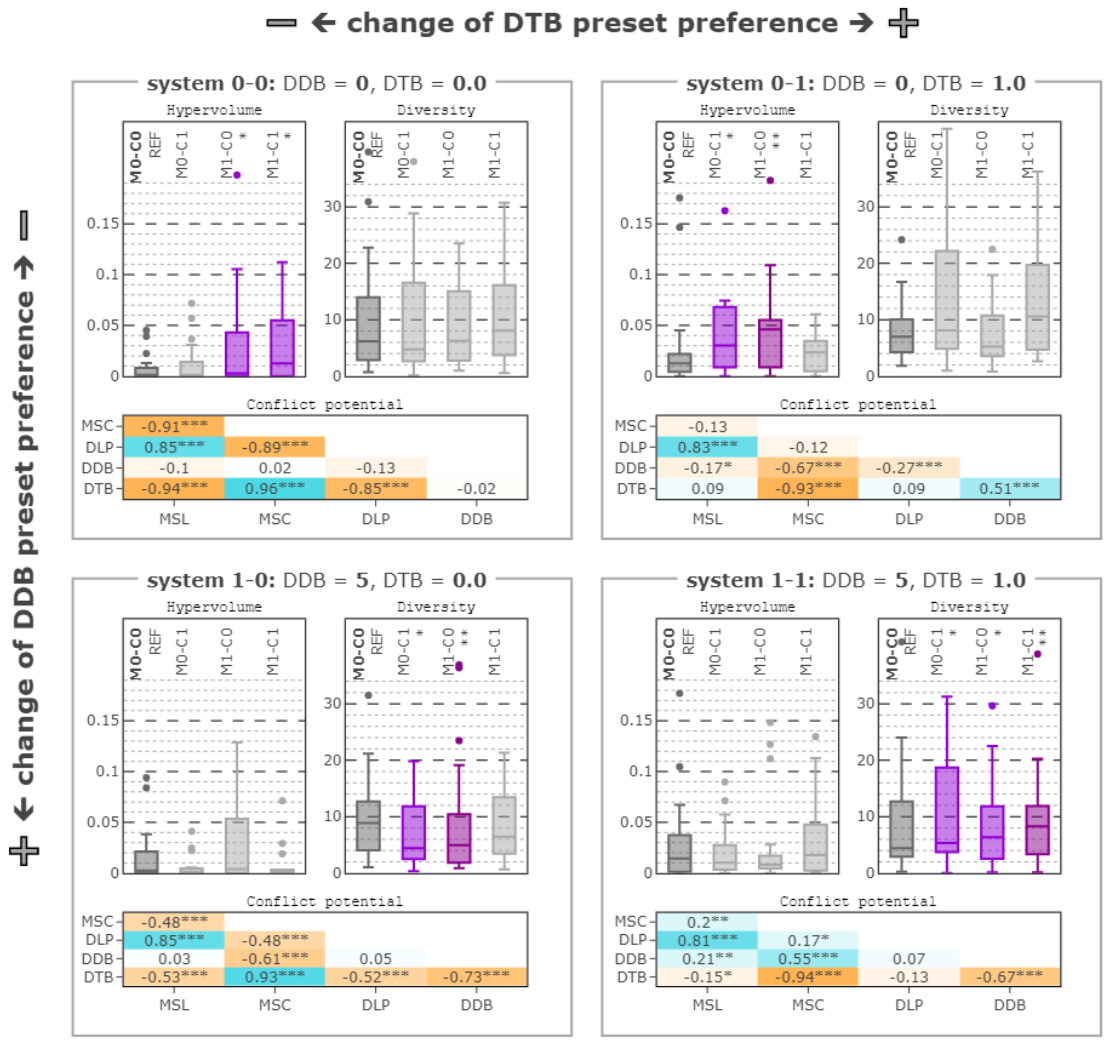

FIGURE B-1: Sensitivity analysis for DLP preset preference 1:6

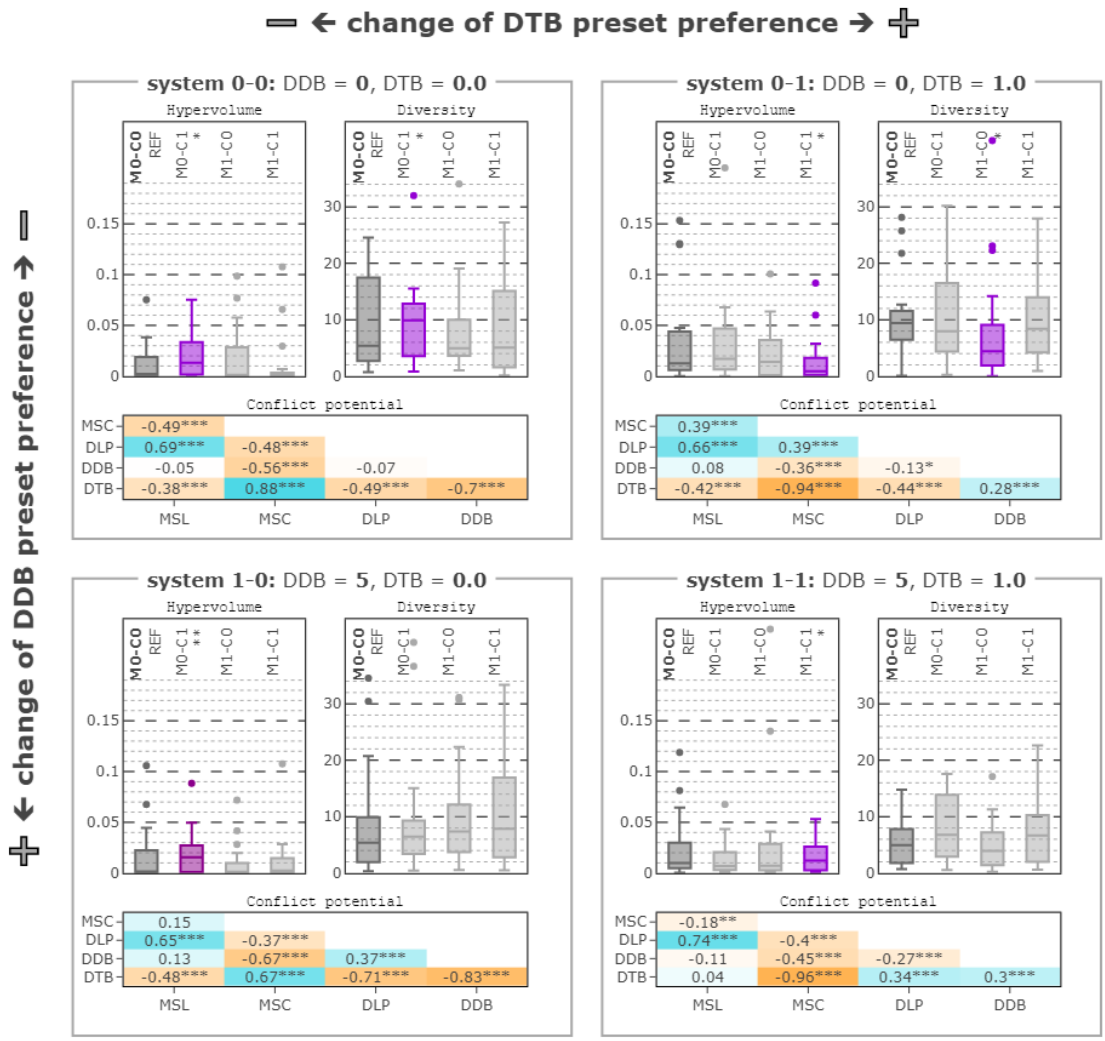

FIGURE B-2: Sensitivity analysis for DLP preset preference 2:5

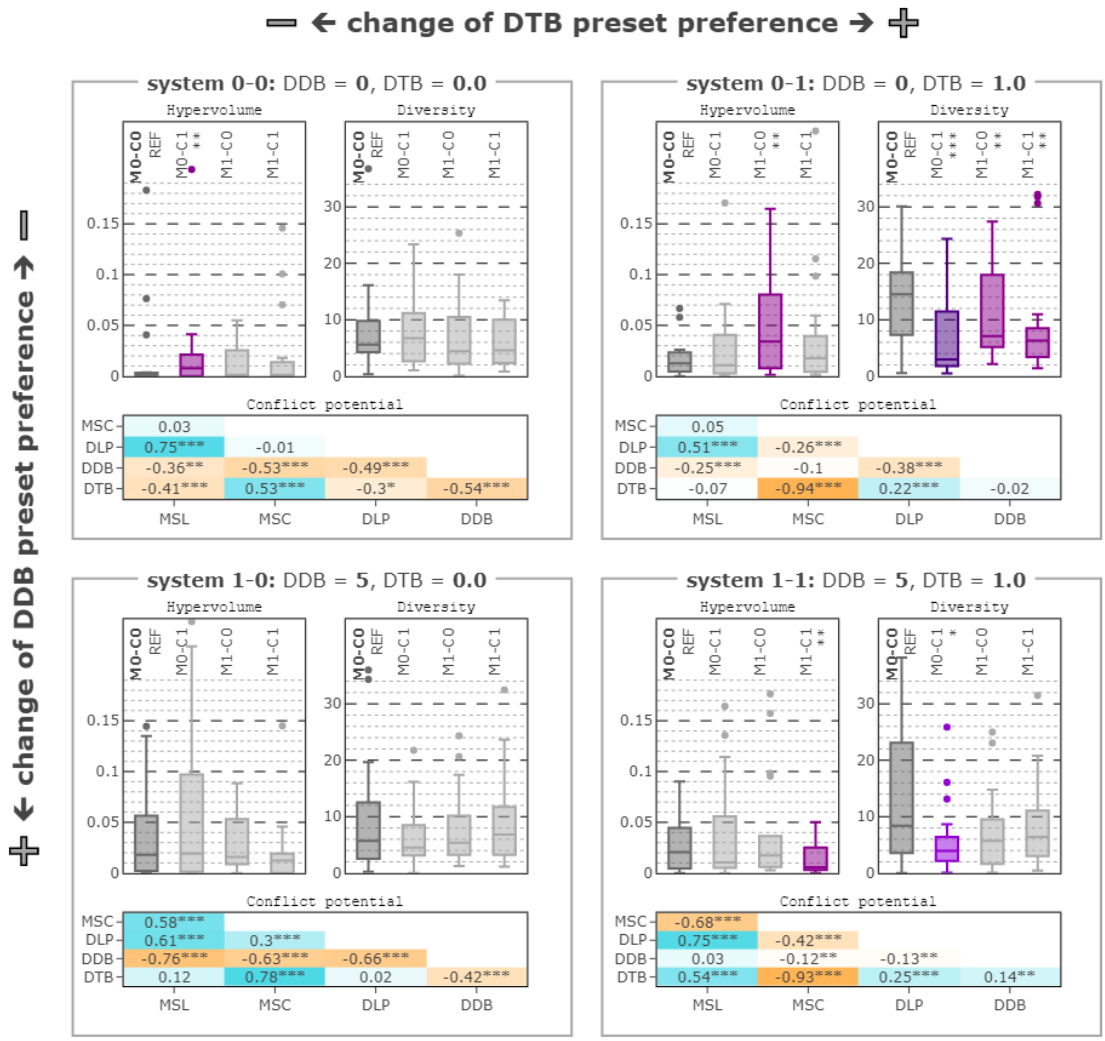

FIGURE B-3: Sensitivity analysis for DLP preset preference 3:4

Bibliography

[1] ‘schedule noun - Oxford Advanced Learner’s Dictionary’. Accessed: Dec. 06, 2022. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/schedule_1

[2] C. Stylianou and A. S. Andreou, ‘Intelligent Software Project Scheduling and Team Staffing with Genetic Algorithms’, in *Artificial Intelligence Applications and Innovations*, Berlin, Heidelberg, 2011, pp. 169–178. doi: 10.1007/978-3-642-23960-1_21.

[3] ‘ShotGrid | (formerly Shotgun) | Autodesk’. https://www.autodesk.com/products/shotgrid/overview (accessed Dec. 06, 2022).

[4] ‘Reclaim | Smart Scheduling for Busy Teams’. https://reclaim.ai/ (accessed Dec. 06, 2022).

[5] ‘History of the Eight Hour Day’, Aug. 22, 2011. Accessed: Dec. 06, 2022. [Online]. Available: https://web.archive.org/web/20110822204743/http://www.8hourday.org.au/pdf/888_fact_01_history.pdf

[6] ‘Busting the attention span myth - BBC News’. Accessed: Dec. 06, 2022. [Online]. Available: https://www.bbc.com/news/health-38896790

[7] J. F. Duffy and C. A. Czeisler, ‘Age-related change in the relationship between circadian period, circadian phase, and diurnal preference in humans’, *Neurosci Lett*, vol. 318, no. 3, pp. 117–120, Feb. 2002, doi: 10.1016/s0304-3940(01)02427-2.

[8] M. A. Carmichael, R. L. Thomson, L. J. Moran, and T. P. Wycherley, ‘The Impact of Menstrual Cycle Phase on Athletes’ Performance: A Narrative Review’, *Int J Environ Res Public Health*, vol. 18, no. 4, p. 1667, Feb. 2021, doi: 10.3390/ijerph18041667.

[9] S. Mostaghim, ‘Swarm Intelligence’, 2021.

[10] E.-G. Talbi, *Metaheuristics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009. doi: 10.1002/9780470496916.

[11] W. Clark, *The Gantt Chart: A Working Tool of Management*. Ronald Press Company, 1922.

[12] J. Sutherland and J. J. Sutherland, *Scrum: The Art of Doing Twice the Work in Half the Time*, Illustrated edition. New York: Currency, 2014.

[13] United States. Bureau of Naval Weapons. Special Projects Office, *Program evaluation research task (PERT) summary report; phase 1*. 1958.

[14] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, 'Chapter 9 Sequencing and scheduling: Algorithms and complexity', in *Handbooks in Operations Research and Management Science*, vol. 4, Elsevier, 1993, pp. 445–522. doi: 10.1016/S0927-0507(05)80189-6.

[15] B. Tan, H. Ma, and Y. Mei, 'A NSGA-II-based Approach for Multi-objective Micro-service Allocation in Container-based Clouds', in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, May 2020, pp. 282–289. doi: 10.1109/CCGrid49817.2020.00-65.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990.

[17] S. Martello, 'Bin packing problems', 2019.

[18] A. Holder and H. J. Greenberg, Eds., *Harvey J. Greenberg: a legacy bridging operations research and computing*. Cham, Switzerland: Springer, 2021. doi: 10.1007/978-3-030-56429-2.

[19] C. M. Fonseca and P. J. Fleming, 'Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation', *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, Jan. 1998, doi: 10.1109/3468.650319.

[20] *Feasibility and Infeasibility in Optimization*, vol. 118. Boston, MA: Springer US, 2008. doi: 10.1007/978-0-387-74932-7.

[21] J. Branke, *Multiobjective Optimization: interactive and evolutionary approaches*. Berlin: Springer-Verlag, 2008.

[22] H. Ishibuchi, R. Imada, N. Masuyama, and Y. Nojima, 'Comparison of Hypervolume, IGD and IGD+ from the Viewpoint of Optimal Distributions of Solutions', in *Evolutionary Multi-Criterion Optimization*, vol. 11411, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham: Springer International Publishing, 2019, pp. 332–345. doi: 10.1007/978-3-030-12598-1_27.

[23] S. Mostaghim, 'Evolutionary Multi-Objective Optimization', 2021.

[24] F. Glover, 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, Jan. 1986, doi: 10.1016/0305-0548(86)90048-1.

[25] M. Mitchell, 'Genetic algorithms: An overview', *Complexity*, vol. 1, no. 1, pp. 31–39, 1995, doi: 10.1002/cplx.6130010108.

[26] S. Poles, Y. Fu, and E. Rigoni, ‘The Effect of Initial Population Sampling on the Convergence of Multi-Objective Genetic Algorithms’, in *Multiobjective Programming and Goal Programming*, vol. 618, V. Barichard, M. Ehrgott, X. Gandibleux, and V. T’Kindt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 123–133. doi: 10.1007/978-3-540-85646-7_12.

[27] A. H. Wright, ‘Genetic Algorithms for Real Parameter Optimization’, in *Foundations of Genetic Algorithms*, vol. 1, G. J. E. Rawlins, Ed. Elsevier, 1991, pp. 205–218. doi: 10.1016/B978-0-08-050684-5.50016-1.

[28] F. P. Brooks, *The mythical man-month: essays on software engineering*, Anniversary ed. Reading, Mass: Addison-Wesley Pub. Co, 1995.

[29] ‘Human-centred design processes for interactive systems (DIN EN ISO 13407)’. Accessed: Dec. 06, 2022. [Online]. Available: https://www.beuth.de/de/norm/din-en-iso-13407/26379257

[30] G. Banjac, P. Goulart, B. Stellato, and S. Boyd, ‘Infeasibility Detection in the Alternating Direction Method of Multipliers for Convex Optimization’, in *2018 UKACC 12th International Conference on Control (CONTROL)*, Sep. 2018, pp. 340–340. doi: 10.1109/CONTROL.2018.8516858.

[31] S. Zilberstein, ‘Using Anytime Algorithms in Intelligent Systems’, *AI Magazine*, vol. 17, no. 3, Art. no. 3, Mar. 1996, doi: 10.1609/aimag.v17i3.1232.

[32] A. Munir, A. Gordon-Ross, and S. Ranka, ‘Optimization Approaches in Distributed Embedded Wireless Sensor Networks*’, in *Modeling and Optimization of Parallel and Distributed Embedded Systems*, IEEE, 2016, pp. 141–158. doi: 10.1002/9781119086383.ch6.

[33] *The Second Brain - A Life-Changing Productivity System*, (Aug. 20, 2020). Accessed: Dec. 04, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=OP3dA2GcAh8

[34] V. I. Levenshtein, ‘Двоичные коды с исправлением выпадений, вставок и замещений символов (binary codes capable of correcting deletions, insertions, and reversals)’, *Доклады Академий Наук СССР*, vol. 163, no. 4, pp. 845–848, 1965.

[35] A. Damci and G. Polat, ‘Impacts of different objective functions on resource leveling in construction projects: A case study’, *Journal of Civil Engineering and Management*, vol. 20, Jul. 2014, doi: 10.3846/13923730.2013.801909.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, ‘A fast and elitist multiobjective genetic algorithm: NSGA-II’, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[37] J. Blank, K. Deb, and P. C. Roy, ‘Investigating the Normalization Procedure of NSGA-III’, in *Evolutionary Multi-Criterion Optimization*, vol. 11411, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P.

Reed, Eds. Cham: Springer International Publishing, 2019, pp. 229–240. doi: 10.1007/978-3-030-12598-1_19.

[38] E. Gamma, Ed., *Design patterns: elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley, 1995.

[39] J. Glamsch, T. Rosnitschek, and F. Rieg, 'Initial Population Influence on Hypervolume Convergence of NSGA-III', *Int. j. simul. model.*, vol. 20, no. 1, pp. 123–133, Mar. 2021, doi: 10.2507/IJSIMM20-1-549.

[40] J. Hugosson, E. Hemberg, A. Brabazon, and M. Neill, 'An investigation of the mutation operator using different representations in grammatical evolution', *Applied Soft Computing - ASC*, pp. 409–419, Jan. 2007.

[41] A. Hassanat, E. Alkafaween, N. Alnawaiseh, M. Abbadi, M. Alkasassbeh, and M. Alhasanat, 'Enhancing genetic algorithms using multi mutations: Experimental results on the travelling salesman problem', *International Journal of Computer Science and Information Security*, vol. 14, pp. 785–801, Sep. 2016.

[42] J. P. Pabico and E. A. Albacea, 'The Interactive Effects of Operators and Parameters to GA Performance Under Different Problem Sizes'. arXiv, Aug. 01, 2015. Accessed: Nov. 08, 2022. [Online]. Available: http://arxiv.org/abs/1508.00097

[43] K. Deep and H. Mebrathu, 'Combined mutation operators of genetic algorithm for the travelling salesman problem', *Int. J. Comb. Optim. Probl. Informatics*, vol. 2, pp. 2–24, 2011.

[44] F. Wang, 'A successive domain-reduction scheme for linearly constrained quadratic integer programming problems', *Journal of the Operational Research Society*, vol. 72, no. 10, pp. 2317–2330, Oct. 2021, doi: 10.1080/01605682.2020.1784047.

[45] R. Kolisch and A. Sprecher, 'PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program', *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, Jan. 1997, doi: 10.1016/S0377-2217(96)00170-1.

[46] A. Scholl, R. Klein, and C. Jürgens, 'Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem', *Computers & Operations Research*, vol. 24, no. 7, pp. 627–645, Jul. 1997, doi: 10.1016/S0305-0548(96)00082-2.

[47] J. Blank and K. Deb, 'Pymoo: Multi-Objective Optimization in Python', *IEEE Access*, vol. 8, pp. 89497–89509, 2020, doi: 10.1109/ACCESS.2020.2990567.

[48] The Danish Pan-Genome Consortium, J. A. Sibbesen, L. Maretty, and A. Krogh, 'Accurate genotyping across variant classes and lengths using variant graphs', *Nat Genet*, vol. 50, no. 7, pp. 1054–1059, Jul. 2018, doi: 10.1038/s41588-018-0145-5.

[49] B. Tan, H. Ma, and Y. Mei, ‘A NSGA-II-based Approach for Multi-objective Micro-service Allocation in Container-based Clouds’, in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, May 2020, pp. 282–289. doi: 10.1109/CCGrid49817.2020.00-65.

[50] J. Blank and K. Deb, ‘A Running Performance Metric and Termination Criterion for Evaluating Evolutionary Multi- and Many-objective Optimization Algorithms’, in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Glasgow, United Kingdom, Jul. 2020, pp. 1–8. doi: 10.1109/CEC48606.2020.9185546.

[51] G. D. Ruxton, ‘The unequal variance t-test is an underused alternative to Student’s t-test and the Mann–Whitney U test’, *Behavioral Ecology*, vol. 17, no. 4, pp. 688–690, Jul. 2006, doi: 10.1093/beheco/ark016.

[52] ‘What do the stars next to the p-values mean? – Council of Europe’. https://faq.edqm.eu/pages/viewpage.action?pageId=1377305 (accessed Dec. 06, 2022).

[53] Y. Cao, B. J. Smucker, and T. J. Robinson, ‘On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design’, *Journal of Statistical Planning and Inference*, vol. 160, pp. 60–74, May 2015, doi: 10.1016/j.jspi.2014.12.004.

[54] C. M. Fonseca, K. Klamroth, G. Rudolph, and M. M. Wiecek, ‘Scalability in multiobjective optimization’, *Dagstuhl Reports*, vol. 10, no. 1, pp. 52–129, 2020, doi: 10.4230/DagRep.10.1.52.

[55] R. Moritz, E. Reich, M. Bernt, and M. Middendorf, ‘The Influence of Correlated Objectives on Different Types of P-ACO Algorithms’, Apr. 2014. doi: 10.1007/978-3-662-44320-0_20.

[56] M. M. Mukaka, ‘Statistics corner: A guide to appropriate use of correlation coefficient in medical research’, *Malawi Med J*, vol. 24, no. 3, pp. 69–71, Sep. 2012.

[57] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima, ‘Modified Distance Calculation in Generational Distance and Inverted Generational Distance’, in *Evolutionary Multi-Criterion Optimization*, vol. 9019, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, Eds. Cham: Springer International Publishing, 2015, pp. 110–125. doi: 10.1007/978-3-319-15892-1_8.

[58] M. Fleischer, ‘The Measure of Pareto Optima Applications to Multi-objective Metaheuristics’, in *Evolutionary Multi-Criterion Optimization*, vol. 2632, C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, and K. Deb, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 519–533. doi: 10.1007/3-540-36970-8_37.

[59] H. K. Singh, ‘Understanding Hypervolume Behavior Theoretically for Benchmarking in Evolutionary Multi/ Many-objective Optimization’, *IEEE Trans. Evol. Computat.*, pp. 1–1, 2019, doi: 10.1109/TEVC.2019.2931191.

[60] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, ‘Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications’, *Theoretical Computer Science*, vol. 425, pp. 75–103, Mar. 2012, doi: 10.1016/j.tcs.2011.03.012.

[61] C. R. B. Azevedo and A. F. R. Araujo, ‘Correlation between diversity and hypervolume in evolutionary multiobjective optimization’, in *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, USA, Jun. 2011, pp. 2743–2750. doi: 10.1109/CEC.2011.5949962.

[62] R. W. Morrison and K. A. De Jong, ‘Measurement of Population Diversity’, in *Artificial Evolution*, vol. 2310, P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 31–41. doi: 10.1007/3-540-46033-0_3.