UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Sander Jenk

A multi-objective optimizer to retrieve issue reports based on developer experience and business value

Master's Thesis (30 ECTS)

Supervisor: Ezequiel Scott, PhD

Tartu 2022

Contents

A multi-objective optimizer to retrieve issue reports based on developer experience and business value

Abstract:
In Agile Software Development, software gets delivered in short iterations. Selecting work for an iteration is complex for multiple reasons. When planning the iteration, developers need to consider their experience, preferences, and work capacity while maximizing the business value. To do this, developers have to understand the content of the issue reports, which is time-consuming because the backlogs can contain thousands of issues. With these things in mind, an automatic multi-objective approach is proposed in this thesis that retrieves issues from the backlog for a developer based on their work capacity and optimizes for the business value of the issue, developer's previous experience with similar issues, and the novelty of the issue. The approach uses LDA to extract topics from the issues. These topics are used to define the developer experience and novelty. NSGA-II is used as the optimization algorithm to extract the set of issues that satisfy the 3 objectives. The approach is evaluated using the data of 15 open-source projects and 1 closed-source project. The evaluation includes an analysis of execution times and the quality of the solutions based on Hypervolume. In addition, a survey with developers is conducted to better understand their opinion and the quality of the solutions. The results show that you can get optimal solutions in less than 4 seconds on average, which is considerably better than the time developers take to manually select issue reports under the same conditions. The answers from the survey show positive results since the approach optimizes for the 3 selected objectives. For these reasons, the tool will be beneficial in the sprint planning process of software projects.

Keywords:
Multi-objective optimization, natural language processing, agile software development

CERCS: P170 Computer science, numerical analysis, systems, control

Mitme eesmärgiga optimeerija leidmaks tööülesandeid vastavalt tarkvaraarendaja kogemusele ja ärilisele väärtusele

Lühikokkuvõte:

Välearendust kasutavates tarkvaraprojektides tarnitakse tarkvara lühikeste iteratsioonidena. Tööülesannete valimine iteratsiooni on keeruline mitmel põhjusel. Iteratsiooni planeerides peavad tarkvaraarendajad arvesse võtma enda kogemust, eelistusi ja töömahtu ning maksimeerima iteratsiooni ärilist väärtust. Selleks peavad nad olema tuttavad ülesannete sisuga, mis on aeganõudev, kuna tegemata tööde prioriteediloend võib olla tuhandeid kirjeid pikk. Neid faktoreid arvestades on selle magistritöö eesmärk välja töötada automaatne mitme eesmärgiga optimeerimismeetod, mis leiaks tegemata tööde loendist ülesanded vastavalt tarkvaraarendaja töövõimakusele ja maksimeeriks kolme eesmärki: tööülesannete ärilist väärtust, arendaja eelnevat kogemust sarnaste tööülesannetega ja tööülesande uudsust arendaja vaatepunktist. Välja töötatud meetod kasutab LDA algoritmi tööülesannetest teemade ekstraheerimiseks ja neid teemasid kasutatakse arendaja eelneva kogemuse ja tööülesande uudsuse defineerimiseks. NSGA-II algoritmi kasutatakse kolme eesmärgi optimeerimiseks, et leida parimad tööülesannete kombinatsioonid tegemata tööde loendist. Välja töötatud meetodi efektiivsust näidatakse kasutades 15 avatud lähtekoodiga projekti ja 1 kinnise lähtekoodiga projekti andmestikke. Meetodi efektiivsuse hindamine sisaldab käitusaja uurimist ja leitud lahendite kvaliteedi analüüsi kasutades Hypervolume sooritusnäitajat. Lisaks küsitletakse tarkvaraarendajaid, et paremini mõista nende eelistusi ja genereeritud lahendite kvaliteeti. Tulemused näitavad, et optimeeritud lahendite genereerimine võtab keskmiselt alla 4 sekundi, mis on oluliselt kiirem võrreldes manuaalselt tööülesannete valimisega samu kriteeriumeid arvesse võttes. Küsitlus näitab samuti positiivseid tulemusi, kuna meetod optimeerib 3 eesmärki arvesse võttes. Välja töötatud meetod on neid tulemusi arvesse võttes iteratsioonide planeerimiseks kasulik tööriist.

Võtmesõnad:

Mitme eesmärgiga optimeerimine, loomuliku keele töötlus, välearendus

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

1 Introduction

In Agile Software Development, software gets delivered in short iterations that represent small increments. Some agile methodologies such as Scrum prescribe planning phases where work that should be completed during the iteration is selected [22]. This task is not trivial for multiple reasons. First, it has been shown that work items to be done in a software project grow over time and can reach thousands [14]. When developers need to plan the work items to be completed during the next iteration, they have to go over their descriptions and understand their content, which is a time-consuming activity and requires knowledge about the business domain and the systems being developed. Moreover, if the product backlog is large, there are many combinations of work items that could be selected for the iteration, and exploring all the possible combinations is also time-consuming. Second, the goal of an iteration is to deliver as much business value as possible [22]. For example, urgently needed new functionality or a critical bug that prevents users from using the system should be planned into the iteration before more trivial work items. Third, the amount of work to be added into an iteration is limited by the amount of work that the team can do. Choosing too many tasks means that all work may not be completed and choosing too few means that team members could be idling at the end of the sprint.

In addition, in open-source projects where work self-assignment is in place, developers have been shown to choose work items based on multiple factors [16]. Among them are having previous experience with similar issues in the past and the opportunity to learn new technology, tools, and domains. Issues, that match developer preferences should therefore be planned into sprints. The goal is to take all of the aforementioned aspects into consideration when choosing issues for a sprint, which makes it a multi-objective optimization problem. Business value and developer preferences should be maximized, while the quantity of work should remain doable. Due to the large backlogs and time-consuming nature of sprint planning, the use of an automatic tool to support sprint planning could make the software development of an agile team easier.

The aim of this thesis is to simplify the sprint planning activity by introducing an approach that automatically retrieves a set of issues from the backlog for a developer. The approach is based on a multi-objective algorithm that optimizes for business value

delivered, the developer's experience with similar issues in the past, and the novelty of the issue from the perspective of the developer. To validate the approach, quantitative and qualitative methods are used. The quantitative evaluation includes an experiment that uses data from open-source software projects and calculates Hypervolume. The qualitative evaluation analyzes the results of a survey conducted on real-life developers, where they are asked to evaluate the sprint plan generated by the approach described in this thesis.

This thesis is structured as follows: Section 2 shows an overview of Agile Software Development in the context of selecting work items for the next iteration. Section 3 provides an overview of related research. Section 4 describes the methodology followed in this thesis. The results are presented in section 5 and further discussed along with the limitations of the thesis in section 6. In section 7, I give a conclusion to the thesis and propose future work.

2 Background

2.1 Agile software development

Agile software development is a practice, where the software gets delivered early and continuously [17]. This is a stark contrast to earlier models like the waterfall model, where all software development activities are completed linearly and sequentially, meaning that only when one activity is completely finished, the project proceeds to the next software development activity [27]. Activities like requirements engineering, system design, development, testing, and deployment are completed once and in order. This model doesn't allow for changing requirements. In Scrum software gets delivered often, usually in 2-4 week increments, which are called sprints. All aforementioned activities related to software development are done within the same increment in agile software development. This requires cross-functional teams, meaning teams are composed of different roles, like the developer(s), tester(s), operations engineer(s), etc [17]. Cross-functional teams should have all the skills and knowledge to be responsible for delivering their part of the product from beginning to end during the sprint. The outcome of each sprint is a piece of working software. This approach promotes sustainable development and the software team can indefinitely maintain a constant pace.

2.2 Issue reports

In agile projects, the product backlog contains a list of things to be done in a project [1]. Issue tracking systems (e.g. JIRA[1]) are usually used to keep track of work in projects. Within these systems, issue reports are the work items. Issues have many properties related to them, like textual summary, textual description, type, estimation, priority, status, assignee, and the creation date and completion date. Most systems allow for adding custom fields. The summary is one sentence long and contains the most important information about the task. The textual description contains a detailed overview of what needs to be done in the context of this issue. The types are usually predefined and the most common ones are feature requests and bug reports. Priority shows how important is the issue from the client's perspective. It can be represented numerically where a larger

[1] https://www.atlassian.com/software/jira

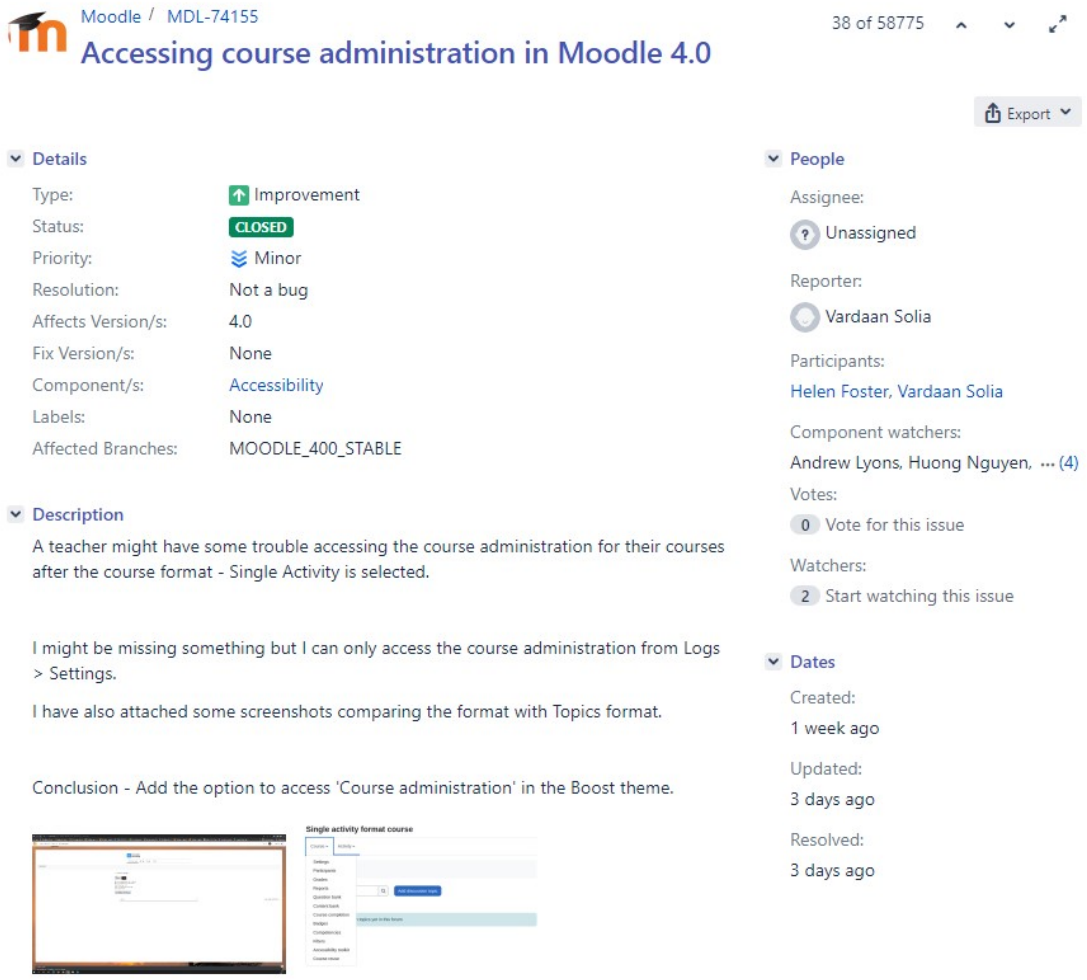

Figure 1. An example of a JIRA issue

value corresponds with higher priority and a smaller value with lower priority. Estimation is usually represented as story points. Story points are relative units of measure, that show how much effort is required to complete the issue. Relative means that an issue with 2 story point estimation requires twice the amount of work when compared to an issue with 1 story point estimation.

2.3 Sprint planning

Scrum is one of the most popular agile software development frameworks. It defines the activities that facilitate the development process. One of these events is sprint planning, where it is decided what will be done in the next sprint [22]. During sprint planning, the sprint goal is defined. The sprint goal expresses the business value delivered during the upcoming sprint. The team then decides which work items match the sprint goal and then selects them for the sprint. It is important to account for the team's capability when choosing the set of issues to be included in the sprint [22]. When choosing too few issues, then the work could be completed before the end of the sprint and cause idle time for team members and additional resources are needed to select more tasks for the sprint. When choosing too many issues, then the work might not get done during the sprint. The capability of the team is measured with velocity [19]. Velocity is calculated by taking the average of story points completed by the team in previous sprints. Velocity therefore can be used to estimate the number of story points the team is able to complete in the upcoming sprint. The sum of story points of the issues selected in the sprint should match the team velocity as close as possible. Some software projects don't use story points, which means that the number of issues represents the velocity in these projects.

2.4 Issue self-assignment

Self-assignment is a practice used by agile self-organizing teams, where individuals choose to work for themselves [15]. It is used less than other agile practices. While the concept is simple, it is difficult to take it into practice because of different challenges related to the relationships between the team and the individual traits of team members. In some cases, software team managers have insufficient trust in their team members to choose and complete the required work or self-assign tasks that have a higher priority or business value. In some cases, developers lack the confidence to pick tasks themselves. Self-assignment is also more difficult for new team members. It has been shown that self-assignment is more popular in open-source projects since they work differently compared to corporate closed-source projects. Open-source projects usually have a small number of main contributors and a larger amount of volunteer developers contributing in parts of interest. Self-assignment is the most popular task allocation method in open

source projects and it has been observed that micro-management or task delegation through project managers is not prevalent [15].

2.5 Importance of accounting for developer preferences

Lately, there has been a lot of research into employee well-being and its effect on employee performance [7, 8]. The happy/productive worker thesis has been around since the 1920s and more recently the effect on the happiness and unhappiness of employees has been studied in the context of software development. Studies have shown that unhappiness in developers has multiple negative consequences: low cognitive performance and reduced motivation of the developer, lower code quality, and discharging code (deleting parts of code or whole repositories) [7]. Also, unhappiness causes developers to be less productive, causing more delays in projects, that stem from glitches in communication activities and disorganized workflow. All these factors mean, that it is in the interest of software companies and software team's interest to keep their developers happy. While it is not feasible for a software company to influence factors that are not related to work and have an effect on employee well-being, it is possible to influence developer well-being with processes and tasks related to software development.

Developers have noted that they are happier and more motivated when they can assign issues to themselves [15]. This is because they have some level of ownership of these tasks and they see value in what they are doing. Low quality of work was described as the outcome of someone else assigning tasks since usually managers were not aware of the assignee's skill set.

In projects where self-assignment is in place developers consider a set of motivating factors when assigning issues [16]. Among the developer-based factors are the developer's technical ability to perform the task, previous experience with similar tasks (developer experience), and the opportunity to learn new technology, tools, or domains (novelty). In this research, we will handle two of the latter factors in the sprint planning approach.

2.6 Multi-objective optimization

Multi-objective algorithms have been used to solve many real-life problems in the fields of economy, finance, engineering, and programming [10]. They are useful in cases where multiple objectives need to be considered. The objectives are often conflicting and trade-offs have to be made. An example of an optimization problem is buying a car, where the cost, comfort, power, and fuel consumption are considered. A car with more power has worse fuel consumption and a more comfortable car costs more. Considering all objectives at once is a complex task.

Evolutionary algorithms are popular for solving multi-objective problems [33]. The result of multi-objective optimization is a set of solutions that form the Pareto front, where Pareto efficiency can be observed [21]. It is a state, where it is not possible to improve any of the objectives without making others worse. This means that all solutions in the set are optimal, but the objective values vary between all the individuals. Scalarizing is one way to narrow down the output to a single solution [18].

3 Related work

Tuarob et al. [29] proposed a recommendation algorithm (RECAST) that would recommend suitable team members in different roles for a given software task. The authors proposed 20 different features derived from knowledge graphs, that encoded collaboration history, task similarity, and team members' skills. Task similarity graph was built using Latent Dirichlet Allocation (LDA). The algorithm was used to build topic models of the tasks. To calculate task similarities cosine similarity of the topic distribution vectors (TDV) was used. One of the features that used the data from the task similarity graph was task familiarity, which indicated the team's experience dealing with similar issues in the past. The topic model of the issues of the proposed team members was compared to the topic model of the issue at hand - the issue for which the team is being recommended. Machine learning algorithms were compared for the team-fitness scoring function. The best performing algorithm for this was Random Forest. The max-Logit algorithm was used to find the best team recommendations with regard to the team-fitness function out of a vast number of combinations. Tuarob et al. [29] solved a different problem (team composition for a task) compared to my research, which is about sprint planning. But this thesis will use the methods described in Tuarob et al. [29] research to extract topics and compare developer experience to backlog tasks using LDA and compare topic distribution vectors using cosine similarity.

Multi-objective optimization algorithms have been studied in the context of sprint planning [34]. Al-Zubaidi et al. [34] proposed an algorithm to aid with sprint planning. The multi-objective search-based iteration planning (MOSBIP) algorithm was guided simultaneously by two objectives: maximizing the business value delivered in the sprint and also maximizing the alignment with the iteration's goal. Business value was defined as the product of priority and story points value of the issue. The issue description's alignment with the sprint goal was measured using textual similarity metrics like *term frequency* and *inverse document frequency*. They compared their sprint planning algorithm's output to the actual sprints composed by software teams in different projects using widely used metrics, like precision, recall, and F-measure. Their algorithm significantly outperformed random guessing in all the project datasets. They compared different multi-objective optimization algorithms and found that NSGA-II was the best performer.

They also compared multi-objective optimization algorithms to single-objective ones and found that the multi-objective algorithms performed better. The main difference between this research and Al-Zubaidi et al. [34] is that in this research we will take a more developer-centric approach and consider developer well-being as an important aspect of a software project. In their research, they used LDA to construct sprint goals for sprints that didn't have them explicitly described. In this thesis, LDA is used to extract topics from issues and developer issue completion logs to evaluate developer experience and issue novelty to developers. Also, they compared the output of MOBSIP to actually planned sprints and compared the results. In our case, there is no ground truth since we cannot assume, that these projects considered developer experience when composing sprint plans.

4 Methodology

4.1 Research goal

The aim of this thesis is to propose a multi-objective sprint planning approach, that optimizes for business value delivered during the sprint, the developer's experience with similar issues in the past, and the novelty of the issue. This research aims to answer the following research questions:

- **RQ1** - What is the performance of the multi-objective sprint planning approach, that optimizes for business value, developer experience, and issue novelty?
- **RQ2** - What is the utility of the approach perceived by developers?

4.2 The approach

An overview of the approach is shown in Figure 2. The first step consists of acquiring the data from multiple open source software projects. The data includes issue reports from Jira issue tracking software. The following step includes cleaning and pre-processing the data. Next, the dataset is divided into completed issues and incomplete issues.

To represent the topics related to a project, I apply topic modeling techniques to the description of the issue reports. A topic in topic modeling consists of a set of terms that together suggest a shared theme. In an issue report dataset context, the topics may be related to software components, programming languages, and different parts of the system. LDA is used for topic modeling. Before the topic model is built, LDA-specific preprocessing has to be done. Then, the topic model is built using the completed issues.

The completed issues are grouped by the assigned developer. The topic distribution vector is calculated for each developer by applying the LDA model to the corpus of all the completed issues by that developer. The topic distribution vector represents the experience of the developer related to each of the topics discovered from the project.

The topic distribution vector is also calculated for every backlog issue using the LDA topic model. This vector shows how prevalent each of the project's topics is in the backlog issue.

Since the goal of this approach is to find a set of issues from the backlog that are optimized for business value, developer experience, and issue novelty, these objectives have to be quantified. I define a fitness function for each objective for this purpose. The functions take into account the properties of the issue report, the topic distribution vectors of the issue and the developer, and calculate a single value for each objective. Later, this value is maximized by the optimization algorithm. The business value is based on the priority of the issue. The developer experience and novelty are calculated by comparing the developer TDV and backlog issue TDV to understand how much experience the developer has with the topics of the issue and how novel this issue is from the perspective of the developer, respectively.

The retrieved set of issues should contain an amount of work that represents the usual effort made by the developer. This is calculated based on the issue completion history of the developer and in the approach, it is set as the constraint - the number of issues in the output must not exceed the amount of work the developer can complete.

NSGA-II optimization algorithm is used to find the optimal solutions (Pareto front), where each solution is a set of issues. To narrow the output down to a single best solution, I use an achievement scalarizing function (ASF).

4.3 Dataset

The dataset consists of issue reports of multiple open-source projects and a single closed-source project. Different versions of the open-source dataset have been used in recent studies [23, 25, 24]. Table 2 shows the overview of the projects in the dataset.

4.4 Data preprocessing

4.4.1 Closed-source project

The dataset of the closed-source project used in this thesis is collected from the Asana issue tracking tool. The dataset has to be conformed to JIRA naming rules. This means that additional preprocessing has to be done for this dataset. The following steps are needed:

- Dictionary containing priority information is retrieved from ”custom fields” list of an issue and it’s ”display value” is taken and renamed to ”status.name”.

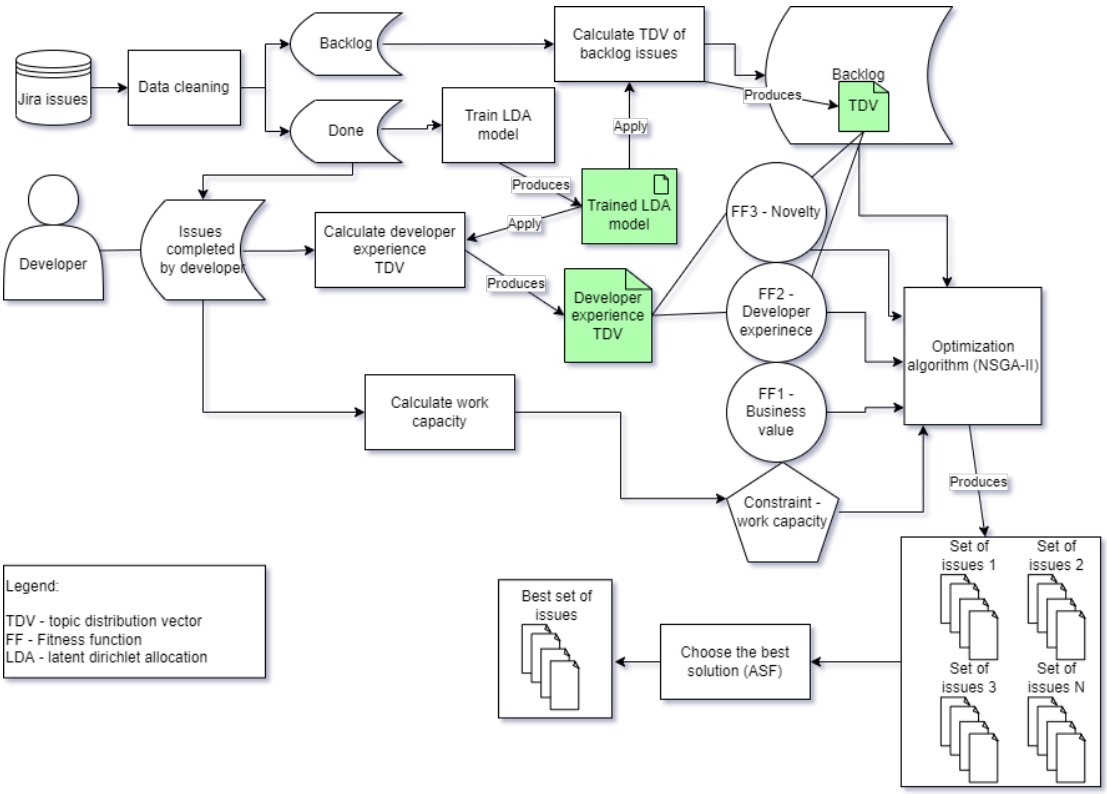

Figure 2. Diagram of approach proposed in this thesis

- ”display value” of ”assignee” dictionary is taken and renamed to ”assignee.name”.
- Based on the ”completed” boolean value ”resolution.name” and ”status.name” are filled. For completed issues resolution ”Done” is assigned. For incomplete issues status ”Backlog” is assigned.
- ”gid” is renamed to ”key”.
- ”name” is renamed to ”summary”.
- ”completed at is renamed to ”resolutiondate”.

After this is done, this dataset is handled uniformly with the rest of the project datasets.

4.4.2 Preprocessing

JIRA software has customizable priority values. Different projects use different priority values and they have to be mapped to numerical values so that the business value could be calculated. For each project, a mapping function is defined. For example, the MOBILE project uses priorities like "Blocker", "Critical", "Major", "Minor", and "Trivial". Project TIMOB uses statuses "Critical", "High", "Low", "Medium", "Trivial" and "None". These values will be mapped to integer values. In the MOBILE example "Trivial" will be mapped to 1 and "Blocker" to 5.

The status and resolution values are also customizable in JIRA, which means different projects have different statuses for done and open issues and different resolutions for done issues. To define done issues all of the corresponding resolutions were collected, which imply doneness. These were across all projects: "Done", "Fixed", "Complete", "Resolved", and "Implemented". Issues that have one of these resolutions are considered done. In order to determine all of the backlog issues similar things had to be done, but with statuses. All statuses that imply the issue being in the backlog were "Open", "To Do", "New", "Backlog", "To Develop", and "Ready for Work". The issues that had a status in that list and do not have an assignee, are considered backlog issues. Issues, that do belong to the backlog or the done issues set, are removed from the dataset. The removed issues include issues in progress or issues that are closed but aren’t solved for some reason etc.

4.4.3 LDA preprocessing

Some LDA-specific preprocessing is also required. First, the textual description and summary will be merged into one field. This will be done because these fields contain all the textual information about the issue. Next, the resulting text is split into words and the words are lower-cased. All punctuation is removed. All stop words and words, that have fewer than 3 characters are removed. Stop words convey no meaning in natural language processing and some examples are "all", "over", "just", "not", "someone", "them". Python library gensim and its list of stopwords are used [20]. Then all the words are lemmatized and stemmed. Stemming reduces words to their original root. This is to make words in multiple forms and tenses uniform. Lemmatization allows grouping together different forms of the same word. For lemmatization and stemming Natural Language Toolkit (nltk) python library is used, more specifically WordNetLemmatizer and SnowballStemmer implementations [2].

4.5 Topic extraction

Topic extraction is a natural language processing (NLP) technique, that is used for discovering abstract topics from a collection of documents [30]. LDA is used for topic extraction in this thesis [4]. It is an unsupervised learning method, which means that the topics are not known beforehand. A topic in topic modeling is a set of terms, that suggest a common theme. For a dataset consisting of issue reports, the topics could be related to software components, tools, domains, and programming languages. The built topic model is used to determine which topics are included in unseen documents. The result of that is a topic distribution vector, where each topic represents a dimension in vector space. Each topic has a value, that represents how prevalent this topic is in a document.

In this thesis, the LDA model is trained on completed issues and is used to determine the topic distribution of every backlog issue. A TDV of a backlog issue shows which topics have to be known by the assigned developer to solve the issue. TDV is also found for the corpus of all issues completed by the developer. This shows how much experience the developer has with each of the topics. These vectors are used to determine the developer experience and novelty objectives.

Gensim's implementation of LDA is used in python. First, dictionaries of words

are built. They will contain each word and the number of times a word appears in the document. The dictionary is then filtered. The words that appear in less than 5 documents are removed along with words that appear in more than 50% of the documents. Then 100000 most frequent words are kept in the dictionary. The next step is to train the LDA model using the dictionary. For this, some hyperparameters need to be defined. The parameter selection is explained in the section 4.7. The resulting model contains topics, which include words and corresponding weights. The weight shows how related the word is to the topic.

4.6 Optimization algorithm overview

For optimization Non-dominated Sorting Genetic Algorithm (NSGA-II) is used [5]. It is a popular, fast, and elitist multi-objective evolutionary algorithm. Elitism strategy ensures that the most fitting individuals are allowed to move to the next generation, without the operators like mutation and crossover worsening their fitness. Elitism can also speed up the performance of genetic algorithms. A python library called pymoo is used as the implementation of the algorithm [3].

4.6.1 Candidate solution

It is important to describe the chromosome or the candidate solution when working with genetic algorithms. The chromosome is represented as a binary list. Since the goal of the thesis is to retrieve a set of issues from the backlog, the list is the length of the backlog, where each list position corresponds with an issue in the backlog. The number 1 means that issue belongs to the candidate solution and 0 means that it doesn't. As an example, if the backlog consists of 5 issues: Issue-1, Issue-2, Issue-3, Issue-4, and Issue-5. A bit list of [0,1,0,1,0] would mean that the solution set consists of 2 issues: Issue-2 and Issue-4.

4.6.2 Fitness functions

NSGA-II requires us to provide the fitness functions that will be used to evaluate the candidate solutions. Each of the objectives has a corresponding fitness function.

The first objective maximizes business value. In this thesis, the business value is defined as the sum of the priority of the issues in the candidate solution. The reasoning

behind it is that in a software project the issue reporter determines the issue priority based on how important it is business-wise.

The second objective aims to maximize developer experience. The developer experience value for a single issue is calculated using the backlog issue TDV and the developer experience TDV. First, topics that the issue doesn't include are discarded. The remaining topics represent the topics that have to be known by the developer to solve the issue. The same topics that were removed from the issue TDV are also removed from the developer experience TDV, since these topics are not relevant - being experienced in these topics do not help with solving the issue. For example when the issue TDV initially looks like [0, 0.2, 0.5] and the developer experience TDV looks like [0.4, 0.2, 0], then after discarding irrelevant topics they look like [0.2, 0.5] and [0.2, 0]. Next, the cosine similarity is calculated between these vectors. A larger cosine similarity means that the developer and issue vectors are more similar and thus the developer has the required skills to complete the issue. In the fitness function, the goal is to maximize the sum of cosine similarities of all issues in the solution set.

The third objective is to maximize issue novelty. Issue novelty is also defined using the issue TDV and the developer experience TDV. The highest topic value is found from the issue vector and the same topic value is checked from the developer experience vector. If this topic value is 0 for the developer, then the issue is considered novel for that developer and the novelty value for this issue is 1. If this topic is higher than 0 in the developer vector, then it is not considered novel and the novelty value is 0. This means that the objective is to maximize the sum of novelty values of the issues in the solution.

4.6.3 Constraint

The retrieved set of issues represents the issues that a developer must complete within a sprint. For this reason, the developer's work capacity must be considered during the sprint planning. The presented approach includes this constraint by limiting the number of retrieved issues. NSGA-II allows defining constraints to discard all candidate solutions, that violate these constraints. The approach proposed in this thesis will have a single constraint - the aforementioned developer work capacity. It will be provided as a parameter to the NSGA-II algorithm. The constraint is defined as follows: the number of issues in a feasible solution can not be higher than the number of issues given

as a parameter. This means that when the estimated number of completed issues for a developer is 5 then the retrieved set of issues must include 5 issues at most.

The estimated number of completed issue reports is calculated as the average number of completed issues in a 2-week time period. Time periods, where the developer was not active (no issues were completed) are not taken into account.

4.6.4 Performance Indicator

Analyzing the performance of the algorithm is critical to understanding how good the solutions are. There are several performance measures used for multi-objective optimization algorithms such as Generational Distance (GD), Generational Distance Plus (GD+), Inverted Generational Distance (IGD), Inverted Generational Distance Plus (IGD+), and Hypervolume (HV) [12, 6]. GD, GD+, IGD, and IGD+ are used to measure the distance from the solution to the Pareto front that represents the most optimal solutions. This obviously means that the Pareto front has to be known for the problem, which is not the case in this thesis. HV doesn't require the Pareto front to be known and instead uses a single user-defined reference point, which means it is a suitable performance measure for the problem described in this thesis.

HV is an indicator that calculates the area/volume from a reference point to possible solutions. In this thesis, 3 objectives are used which means HV is measured in 3-dimensional space. Figure 3 shows HV in the case of 2 objectives. The goal is to maximize HV as a larger value indicates better performance.

HV is widely used because of its multiple properties. First, it is Pareto compliant, which means that maximizing HV guarantees, that the solutions are Pareto optimal [26]. Second, HV can evaluate the convergence and the diversity of the solution simultaneously [26]. Third, one reference point needs to be specified for HV [26].

The reference point for HV is usually defined as a slightly worse point than the Nadir point [11]. The Nadir point specifies the worst possible solution. Due to the implementation of the pymoo package, the objective functions have to be minimized, which means the objective values are either 0 in the worst case or negative. For this reason, the selected reference point in this thesis is [1.1, 1.1, 1.1].

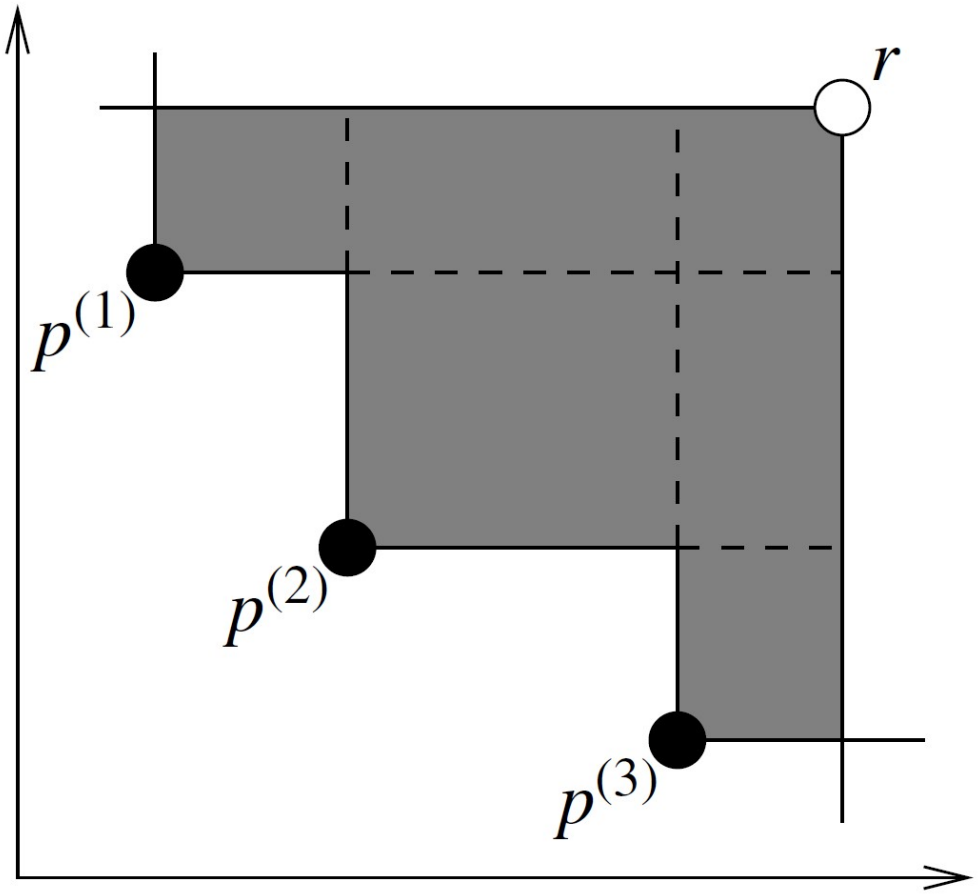

Figure 3. Hypervolume in the case of 2 objectives [6]

4.6.5 Termination criterion

There are different ways to determine when the optimal solutions are found and the execution of the algorithm should be terminated. The simplest way is to define a fixed number of generations. The problem with this approach is that for datasets of different sizes, finding optimal solutions requires a different number of generations. This approach is not suitable because 16 datasets with a different number of issues are used.

Another popular method is to monitor a chosen performance metric during the execution of the algorithm. When the improvement of the metric becomes insignificant, the algorithm is terminated. This is the chosen approach since it works for different dataset sizes. More specifically, after every 10 generations, the HV of the previous 10 generations is analyzed. The minimum and maximum HV are compared and if their difference is less than 0.001, the execution is terminated.

It is important to note that while the HV is a suitable metric to evaluate the convergence of the algorithm, its computation time increases exponentially with the increase of dimensions [26]. This thesis uses 3 objectives, which means HV calculation is still fast. In cases with more objectives, the chosen termination criterion might not be suitable, since HV is calculated after every generation.

I extended pymoo's functionality and implemented the termination criterion chosen for this thesis since it wasn't available by default. The implementation is available in the thesis repository.

4.7 Hyperparameter selection

4.7.1 LDA

LDA requires the definition of multiple hyperparameters - the number of topics, α and η need to be defined [4]. The number of topics determines how many topics will be extracted from the corpus. α controls the document-topic distribution. There are 2 possible distributions: symmetric and asymmetric. For a symmetric distribution, a small α value leads to a document belonging largely to one topic, while a larger value causes the document to include a mix of topics. For asymmetric distribution, higher α causes a more specific topic distribution for a document. The η parameter controls how many words topics contain. When a symmetric distribution is used, topics include more words

with a larger eta value and fewer with a smaller value. Asymmetric distribution causes a more specific word distribution for a topic.

In order to select the hyperparameters for LDA, an experiment is conducted. A range of values is defined for each hyperparameter and for each combination an LDA model is built for each project. To build the model, I took a random sample of 200 issue reports from each project's dataset. I used the coherence score (C_v) to evaluate the performance of each model. C_v is considered as a proxy for topic quality, that shows correlation with human topic ranking [28]. The hyperparameters of the LDA model with the highest coherence score are selected. The values considered for each project are the following:

- Number of topics: 2-10
- α
  - Symmetric: 0.01, 0.31, 0.61, 0.91, 1
  - Asymmetric: fixed normalized asymmetric distribution for each topic that is calculated as $1.0/(TopicIndex + \sqrt{NumberOfTopics})$
- η
  - Symmetric: 0.01, 0.31, 0.61, 0.91, 1

For each project, 270 LDA models were built. Table 1 shows the hyperparameters of the LDA model with the highest C_v for each project, that are selected for the approach.

4.7.2 NSGA-II

In this section, I explain the parameters used for the NSGA-II algorithm. The parameters are selected based on the problem at hand, but they are not tuned in any way, because of the lack of computing resources and time restrictions. Also, the parameters depend on the problem, which means it is not possible to choose the parameters based on previous literature as this problem has not been studied before. Thus reasonable parameters and operators are selected from a set of commonly used values.

NSGA-II is a genetic algorithm and a common way of generating the initial population is creating it randomly [32]. This method is also used in this thesis. More specifically binary random sampling is used since our problem doesn't allow duplicates - it doesn't

Table 1. Best LDA hyperparameters

project	topics	alpha	eta	coherence
compass	5	1	0.31	0.33
datacass	3	0.91	0.31	0.49
fab	10	0.91	0.31	0.37
is	2	asymmetric	0.31	0.39
mdl	8	0.61	0.01	0.32
mobile	3	0.61	0.61	0.68
stl	2	0.01	0.01	0.23
apstud	3	asymmetric	0.91	0.5
dnn	2	asymmetric	0.31	0.51
mesos	5	asymmetric	0.61	0.37
mule	6	0.91	0.31	0.43
nexus	7	0.31	0.61	0.47
timob	10	0.91	0.31	0.56
tistud	9	0.91	0.91	0.63
xd	2	asymmetric	1	0.47
bondora*	8	asymmetric	1	0.40

*Dataset only used for RQ2

make sense to have an issue in the solution set more than once. So each gene on the chromosome will be initialized as either 1 or 0. In pymoo the exact sampling value is "bin_random". 200 individuals are chosen as the initial population size.

Since it is not known which crossover operator is best for the problem in this thesis, one is selected randomly from a set of common crossover operators. Half-uniform crossover is chosen. First, it is determined in which places the parents are different. Then at half of these indices, the values are swapped to produce the offspring. In pymoo crossover value should be "bin_hux".

The mutation operator in genetic algorithms is used to introduce variation into the solutions. Since the goal of the mutation in our context is to randomly include or exclude an issue in an individual to get more diverse result sets the binary bit-flip mutation operator is selected. The probability of a mutation occurring at any point on the individual is selected as 1 divided by the number of issues in the backlog. In pymoo mutation value should be "bin_bitflip".

4.8 Quantitative validation

In order to answer RQ1, an experiment is conducted. A detailed explanation of the experiment:

- Dataset is cleaned and preprocessed. The steps are described in section 4.4.
- Dataset is grouped by project and all the following steps are done for each project separately.
- Project dataset is divided into done and backlog issues.
- Hyperparameters are retrieved for LDA. These are calculated beforehand in an experiment described in section 4.7.
- LDA model is trained using done issues. This is done once per project and the LDA model training time is measured. The time starts when the dataset is preprocessed because in practice this can be done offline. This means that extracting backlog and done issues are included in the time in addition to retrieving the LDA parameters since these steps have to be done for each execution. The time stops when the LDA training is finished.
- Topic distribution vectors are calculated for all backlog issues using the LDA model. Backlog issues are iterated and LDA model is applied to the combined description and summary of each issue.
- All done issues are grouped by the assigned developer, which means that the following steps are done for each developer separately.
- Issues done by the developer are extracted from done issues. Using the developer issues, the developer velocity is calculated. This step is explained in the section 4.6.3.
- The text of all developer issues is combined into a corpus. The developer experience vector is calculated using the trained LDA model and the corpus.

- Issue similarity and novelty are calculated for every backlog issue by comparing the issue topic distribution vector to the developer experience vector. This is described in section 4.6.2.

- Novelty, issue similarity, and business value from the backlog issues are passed to the NSGA-II algorithm as objectives and velocity as the constraint. Parameter selection for the algorithm is explained in section 4.7.2. The results are generated by running the optimization algorithm. The result of the optimization is a non-dominated set of solutions. The time is measured from the optimization execution start time until the results are generated.

- Hypervolume of the non-dominated set of solutions is calculated. The info about this metric is explained in section 4.6.4.

- The project name, developer name, HV of the non-dominated front, LDA execution time, optimization time, and velocity are recorded.

Since the objective functions are defined as sums in this thesis, the recorded HV is higher for a set of solutions that includes more issues. The number of issues in solutions is determined by the issue completion history of the developer. To compare the results of developers and projects, it is necessary to measure the per issue HV of solutions. The average performance of the project is calculated as weighted average - HV of the sets of solutions found for each developer are the items and the number of issues are the weights. This gives us the per-issue performance of the project.

I compare the weighted average HV of the project to the per issue HV of the best solution found in the project and calculate the percentage to evaluate the quality of the solutions found with the approach described in this thesis. This measure represents how good the solutions are on average when compared to the best solution.

In this thesis, a personal laptop with 16 GB of RAM and an Intel(R) Core(TM) i7-9750H processor is used for running all the experiments. The source code of the approach and the results of the experiments are available in the thesis repository[2]. The repository also includes the code for a prototype web app, where the approach can be tested using the open-source datasets used in the experiment.

[2]https://github.com/sanderjenk/thesis

4.9 Qualitative validation

To answer the second research question a survey is conducted with real-life developers. A set of issues that represent a partial sprint plan for each developer is retrieved from a project backlog and a questionnaire is used to collect evaluations on the output of the algorithm. The developers and the dataset are selected based on convenience sampling.

Bondora is a private company with closed-source software systems. The company provides multiple products related to loans and investing. The questionnaire is created for a single software team in the company that includes 5 developers. Other teams do not use issue priorities which means it is not possible to use the approach described in this thesis with the selected business value definition.

The same pipeline is used to generate the sets of issues as in the quantitative analysis for each developer. In the quantitative analysis, the performance of the optimization is measured by evaluating the set of solutions - the whole non-dominated front. In order to find the best solution from the non-dominated front, an additional step is needed. An Achievement Scalarization Function (ASF) is a widely used method in multi-objective optimization to narrow down the set of solutions to a single solution [18]. Pymoo's implementation of that function is used. ASF requires the definition of weights for each objective. In this thesis, the weights are all equal as the goal is to optimize business value, developer experience, and novelty equally. The best set of issues is found for each developer to be used in the survey.

The questionnaire is created in Google Forms. The questions are described in section 7. The first section of the survey includes demographic questions, questions about the preferences of developers when choosing issues, and an estimation of the developer's self velocity.

The second section includes a task for the developers, where they are asked to choose their preferred issues for the upcoming 2-week sprint from a snapshot of their project's backlog. They are required to pick the same amount of issues as the velocity calculated in the previous step (described in section 4.6.3) so that the manually picked issues could be compared to the output of the proposed sprint planning tool. In addition, they are asked how long did it take to select these issues and what was the most difficult part.

The goal of the third section is to let developers evaluate the best set of issues found for them. The first questions are about the accuracy of the estimation. The next questions

ask how prevalent are each of the three objectives in the set of issues with comments from the developers. Finally, it is asked how happy the developer would be with the assigned issues. Survey questions that use the Likert scale are plotted on Likert plots.

The set of issue reports selected by the developers in section 2 is compared to the non-dominated front generated by the optimization algorithm. Generational distance (GD) is used to evaluate how close the hand-picked set of issues is to the calculated Pareto front. The goal is to see how the approach proposed in this thesis rates the hand-picked solution. The hand-picked solution and the best solution along with the rest of the non-dominated front are plotted and displayed in this thesis.

5 Results

5.1 Dataset description

The whole dataset consists of 15 open-source projects and one closed-source project. Table 2 shows general information about the dataset, which includes the project ID, a short description, and the company responsible for the project. The initial dataset consists of 111501 total issues. After pre-processing the dataset size is 75072 issues, which means 36329 issues were removed. MDL is by far the largest project with 64859 issues, 508 developers, and also the longest-running project, where the earliest issue was created in 2002. The next largest project is FAB with 13053 issues. The project with the smallest number of issues is BONDORA, with only 453 total issues. The number of developers ranged from 5 to 508. The issue and developer count for all projects can be seen in Table 3.

Table 3 also shows the number of completed issue reports (Done), the number of issue reports in backlog (Backlog), the date of the first issue (From), and the date of the most recent issue (To)

Table 2. Dataset description

Project ID	Description	Developer
XD	Spring XD	Pivotal Software, Inc.
APSTUD	Aptana studio	Aptana Inc
TISTUD	Appcelerator Studio	Appcelerator Inc
MOBILE	Moodle mobile	Moodle
MDL	Moodle	Moodle
DNN	DNN Platform	DNNSoftware
MESOS	Cluster management software	Apache Software Foundation
MULE	MuleSoft integration platform	MuleSoft
NEXUS	Nexus artifact manager	Sonatype
TIMOB	Titanium Command Line (CLI)	Appcelerator, Inc
COMPASS	MongoDB Compass	MongoDB
FAB	Hyperledger Fabric	Hyperledger
DATACASS	Data Cassandra	Apache Software Foundation
STL	Sawtooth Distributed Ledger	Hyperledger
IS	Indy distributed identity ledger	Hyperledger
BONDORA*	Bondora	Bondora Solutions OÜ

*Closed-source project

Table 3. Number of issues

Project	Total	After preprocessing	Done	Backlog	Devs	From	To
COMPASS	6212	3300	2852	448	12	2016-07-25	2020-03-19
DATACASS	735	605	577	28	10	2013-05-29	2020-03-31
FAB	13053	9414	8860	554	373	2016-07-28	2020-03-13
IS	1516	1164	953	211	79	2017-05-24	2020-04-03
MDL	64859	40919	31606	9313	508	2002-04-25	2020-02-19
MOBILE	3273	2638	2303	335	24	2011-02-14	2020-04-01
STL	3310	3210	2280	930	50	2016-07-18	2020-03-26
APSTUD	886	766	538	228	11	2006-06-05	2015-01-14
DNN	3328	1417	1321	96	11	2012-01-16	2016-04-12
MESOS	2304	1806	1740	66	67	2012-11-19	2016-05-06
MULE	1497	1296	1202	94	35	2011-01-31	2016-05-06
NEXUS	1268	956	914	42	21	2008-08-14	2016-05-05
TIMOB	2144	1847	1724	123	48	2011-04-15	2016-04-20
TISTUD	2879	2557	2386	171	29	2011-03-01	2016-03-29
XD	3784	2983	2361	622	31	2013-04-12	2016-03-31
BONDORA*	453	194	176	18	5	2020-11-23	2022-04-21

*Closed-source project

5.2 Performance results (RQ1)

RQ1 is an inquiry about the performance of the proposed approach. To answer this question, the results of two main aspects are reported:

- the computational time required to find the solutions when the approach is used
- the quality of the solutions found by the approach

Table 4 shows the results grouped by the project. The approach is run once for each developer in the project. The minimum, maximum, standard deviation, and mean weighted HV values are calculated along with the proportion of the mean weighted HV out of the maximum.

Figure 4 shows the weighted average HV of all projects. The solutions with the highest HV on average were found in COMPASS. The weighted average HV of the project is 49.37. The overall best solution was also found for a developer in the COMPASS project with 200.24 per issue HV. On average, the worst solutions were found in the DATACASS project, where the recorded weighted average HV was 4.65. The overall worst solutions were found in DNN and MESOS with 3.7 per issue HV. COMPASS also had the highest

Table 4. Hypervolume of projects

project	Min	Max	Std	Weighted Avg	% from best
COMPASS	11.08	200.24	53.02	49.37	24.65
DATACASS	3.83	9.63	1.77	4.65	48.28
FAB	3.92	100.56	12.46	19.44	19.33
IS	4.15	37.92	4.71	7.53	19.85
MOBILE	4.62	16.06	2.88	7.0	43.56
STL	4.01	37.82	7.16	9.61	25.41
APSTUD	4.72	27.8	7.09	10.68	38.42
DNN	3.7	26.67	7.55	9.44	35.39
MESOS	3.7	49.17	9.22	19.17	38.99
MULE	3.98	14.55	3.86	7.45	51.23
NEXUS	4.27	29.04	6.55	13.7	47.19
TIMOB	4.83	66.64	12.73	24.19	36.31
TISTUD	4.74	54.7	15.0	26.73	48.87
XD	4.35	32.84	8.17	12.35	37.61
BONDORA*	5.7	12.32	2.71	7.88	64.02

*Closed-source project

standard deviation of mean weighted HV (53.02) while the next highest value was 15.0 in the TISTUD project. The lowest standard deviation was 1.77 in the DATACASS project.

An interesting observation to emerge from the data is that solutions with higher HV are found for developers with a higher number of issues as seen in figure 5. The most striking part of the diagram is that it is divided into 2 parts. In one case the HV increases steeply while in the other case the increase is milder. When exploring the 2 extremes in the diagram, it becomes instantly clear why this happens. Both of the cases exist in the COMPASS project. The algorithm finds a lot of novel issues in all the solutions in the Pareto front for the data point with the highest HV. In contrast, no novel issues are found for the data point with 24 issues and relatively low HV value - the other extreme. This happens because the HV measures the volume of the Pareto from the reference point. Novelty is defined in this thesis in a way that if the highest topic value of the issue is 0 on the developer experience vector on the same topic then the issue is novel for the developer. In cases where the developer has some experience in every topic, no issue can be novel. This means that the novelty objective can only be 0 in the objective space and significantly limits the HV value since one dimension is missing. In addition, it can be seen in Figure 6, that projects with a larger number of LDA topics had larger HV values. Having more topics increases the probability that one or more of them is 0 for

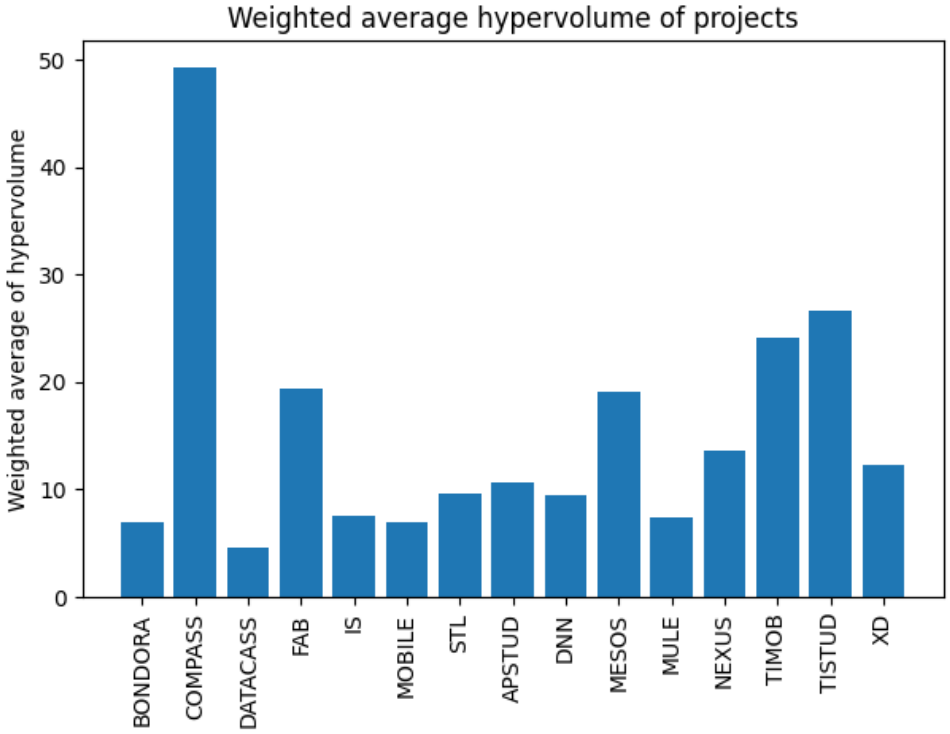

Figure 4. Weighted average HV of projects

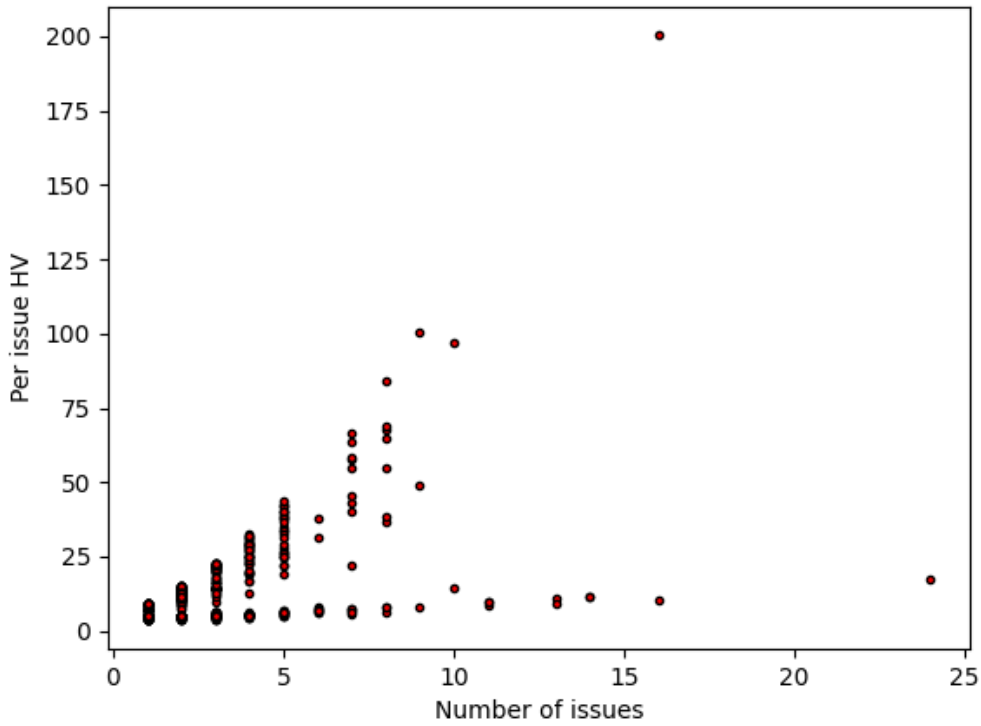

Figure 5. Relation between per issue HV and number of issues

a developer on the experience vector. This increases the HV of the solutions since the solutions with novel issues have significantly higher performance.

What stands out from the data is that a single outlier caused the high average performance of the COMPASS project. The optimization algorithm found solutions with a per-issue HV of 200.24 for one developer, while the next best solution in the project was 67.86. Out of all the projects the next best solution found was in FAB with 100.56, which means the outlier value in COMPASS was almost 2 times better than the next best. The outlier also caused the high standard deviation of the weighted average HV in the COMPASS project and a lower percentage from the best since the best value was so high.

The highest average performance when compared to the highest HV was in the BONDORA project. The weighted average HV was 64.02% from the best solution. The highest percentage among open-source projects was MULE with 51.23%. The lowest performance was observed in the FAB project with 19.33%.

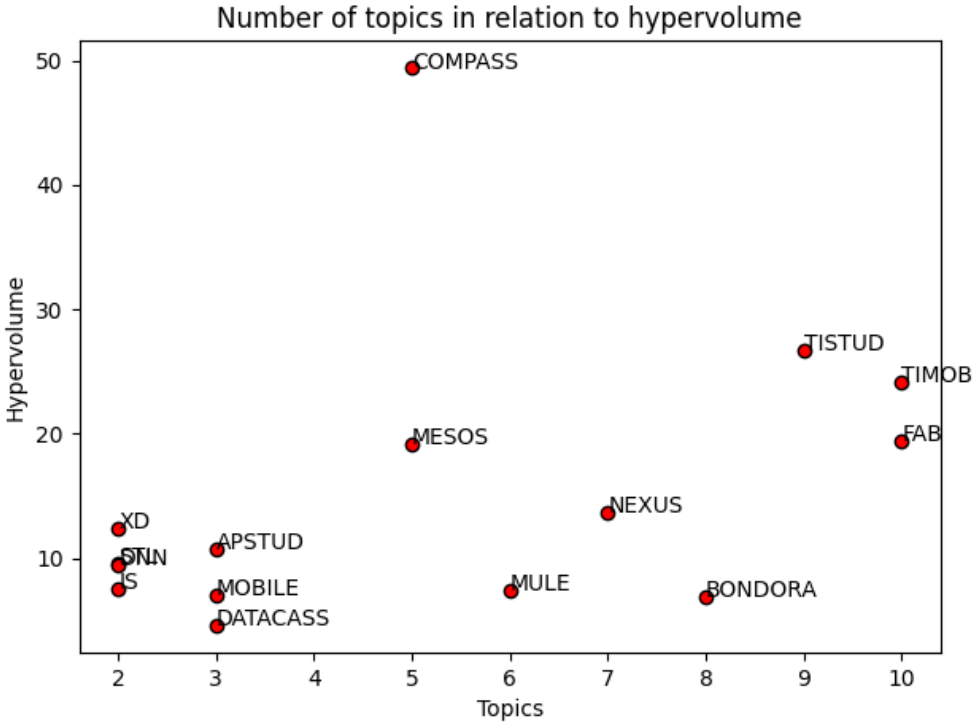

Figure 6. Hypervolume in relation to number of topics

Table 5 shows the execution time of the approach for each project. In each project, the approach is run for each developer and the execution times are aggregated. LDA model training time was the shortest for DATACASS with 3.42 seconds and longest for FAB with 11.99 seconds. These projects also had the smallest and largest number of done issues amongst the open-source projects, respectively. BONDORA the only closed-source project had the lowest amount of done issues but required more time to train than DATACASS. The relationship between the number of done issues and LDA training time can be observed in Figure 7. The lowest mean optimization time was also recorded for the DATACASS project (0.64 seconds) and the highest for STL (7.8 seconds). These projects also had the fewest (28) and most (930) backlog issues. The average optimization time overall was 3.93 seconds. The relation between backlog size and the mean optimization time of the project can be seen in Figure 8.

Table 5. Execution time data

	Execution time in seconds						
Project	LDA	Opt_max	Opt_min	Opt_mean	Opt_std	Backlog	Done
COMPASS	8.68	17.93	2.95	7.41	4.13	448	2852
DATACASS	3.42	1.11	0.32	0.64	0.26	28	577
FAB	11.99	28.99	2.92	5.26	3.0	554	8860
IS	6.94	5.85	1.06	1.75	0.75	211	953
MOBILE	8.19	15.91	2.05	4.86	3.0	335	2303
STL	7.26	11.12	6.05	7.8	1.15	930	2280
APSTUD	6.07	3.74	1.19	2.33	0.87	228	538
DNN	7.58	5.93	0.69	2.76	1.77	96	1321
MESOS	7.98	1.87	0.45	0.82	0.31	66	1740
MULE	6.74	3.93	0.61	1.39	0.73	94	1202
NEXUS	6.9	2.85	0.31	0.87	0.51	42	914
TIMOB	8.79	5.98	0.7	1.87	1.22	123	1724
TISTUD	7.82	11.32	0.98	3.13	3.01	171	2386
XD	7.31	6.81	3.91	5.38	0.76	622	2361
BONDORA*	4.4	0.75	0.61	0.65	0.06	18	176

*Closed-source project

It was also observed that optimization took longer for developers with a larger number of issues. Figure 9 shows the relation between optimization time and the number of issues. An example of higher optimization time because of the higher number of issues is a developer in the COMPASS project with 24 estimated number of issues. While the project's mean optimization time was 7.41, for the developer with large number of issues the optimization took 17.93 seconds.

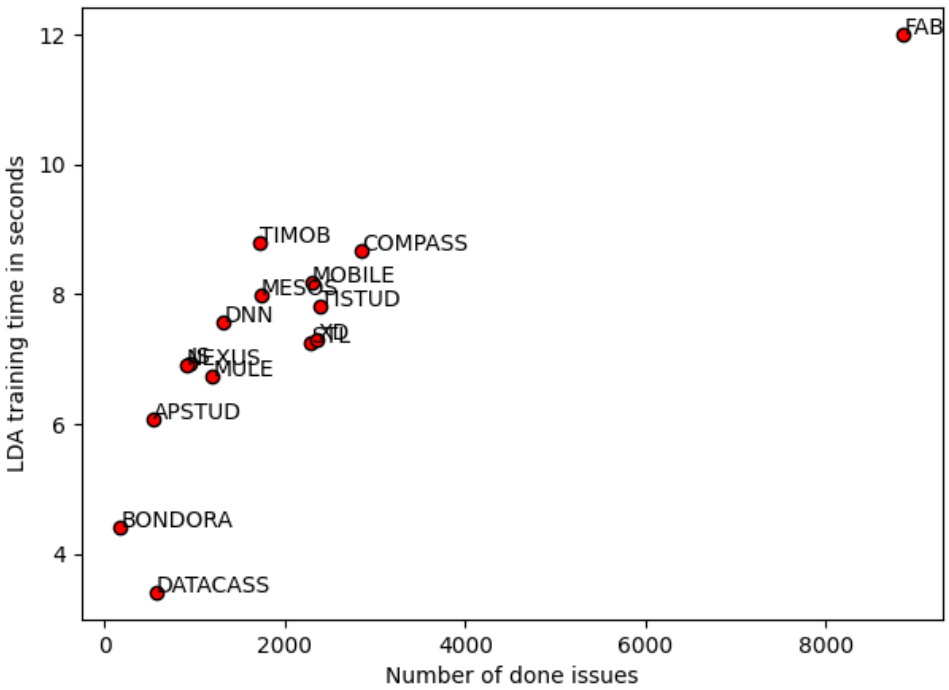

Figure 7. LDA training time in relation to the number of done issues

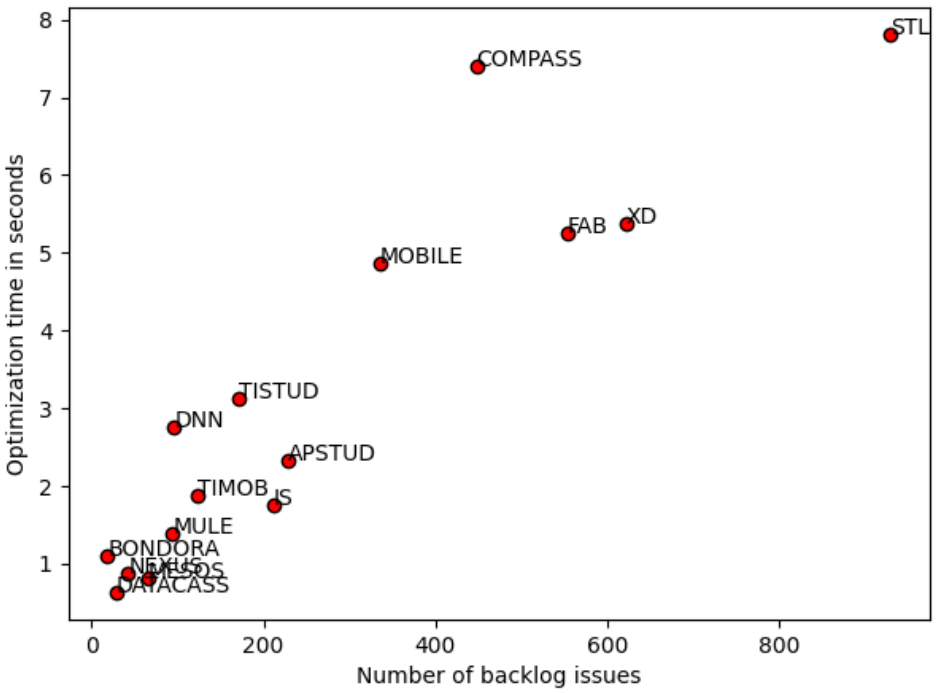

Figure 8. Optimization time in relation to the number of backlog issues

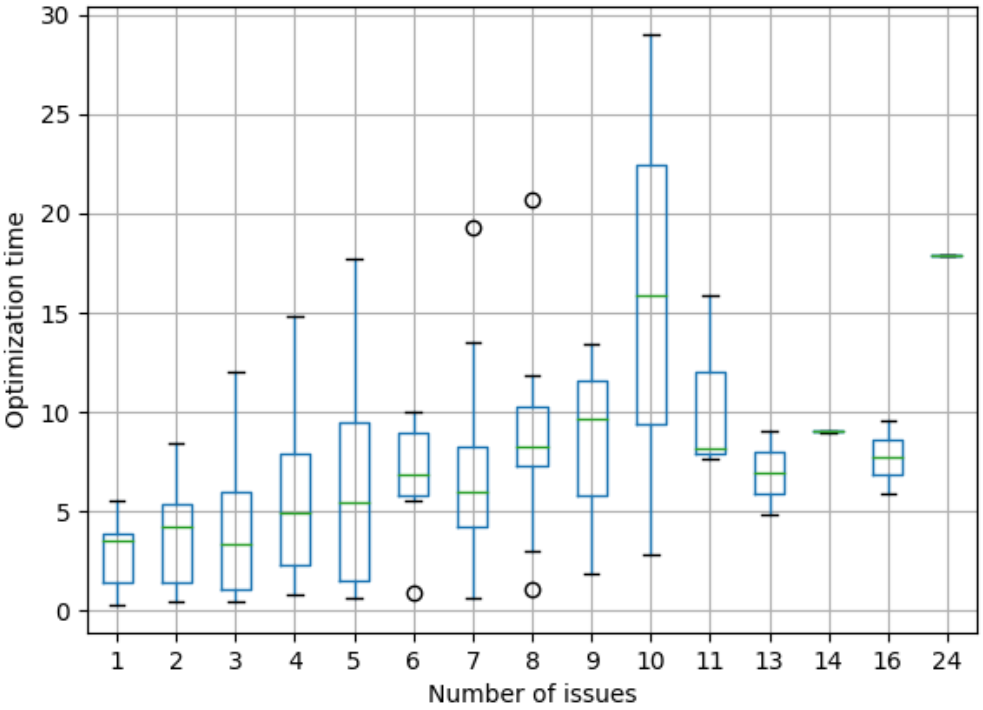

Figure 9. Optimization time in relation to the number of issues.

For the MDL project, it wasn't possible to generate solutions as the optimization algorithm for a single developer did not finish in 20 minutes. The project had a significantly larger backlog compared to other projects. Their backlog included 9313 issues while the next largest was STL with 930 issues, which makes MDL project over 10 times larger. This size difference has a large effect on the performance of the algorithm since the search space is increased significantly and for the algorithm to converge, a large number of generations is needed.

In summary, the software project that uses this approach could expect to get solutions with quality from 20%-75% of the best result (the best HV). It is also difficult to evaluate the solutions based on the HV since the optimal solutions are not known. The LDA training time for software projects with less than 3000 done issues should remain under 9 seconds and for projects with close to 10000 done issues it takes over 12 seconds. In practice, LDA training can be done offline. This means that optimization time is a more relevant performance metric in this approach. For projects with less than a thousand backlog issues, the optimization algorithm finishes in about 8 seconds. With significantly larger backlogs the approach proposed in this thesis might not be viable since the optimization times become longer. Although HV is widely used to evaluate the quality of the solutions in multi-objective optimization algorithms [6, 26], it is also difficult to interpret. For this reason, a qualitative study was conducted to inquire about the quality of solutions generated by the approach from real-life developers.

5.3 Survey results (RQ2)

RQ2 explores the utility of the approach as perceived by developers. A survey is carried out among the developers of a software team. The team is a development unit at an Estonian company, Bondora Solutions OÜ. The developers were asked to evaluate the output of the approach described in this thesis.

Table 6 shows the results of running the approach for each developer in BONDORA project. This is the unaggregated data from the quantitative experiment. For developers of BONDORA additional step was performed. The best solution was calculated from the non-dominated set using ASF and saved to be used in the survey.

All developers involved in the BONDORA project answered the survey. The respondents include a team lead and 4 developers and the respondent's experience varied

Table 6. Bondora performance data

Project	Assignee	HV	Opt time	LDA time	Nr of issues	Picked GD from PF
BONDORA	Developer 1	17.72	0.65	4.4	2	0.6167
BONDORA	Developer 2	17.56	0.66	4.4	2	0.5534
BONDORA	Developer 3	17.10	0.61	4.4	3	0.7884
BONDORA	Developer 4	17.59	0.75	4.4	3	0.9725
BONDORA	Developer 5	24.63	0.61	4.4	2	1.4687

between 1 and 7 years in their current roles. In section 1 of the survey, the developers were asked how much they consider the 3 objectives of this thesis when self-assigning issues. All developers answered that they take business value, experience with similar issues in the past, and issue novelty into account - none of them answered with 1 on the Likert scale. The Likert plot with these answers can be seen in Figure 10. The figure also shows that the developers consider these 3 objectives equally important. Developers like working on issue reports related to topics that they have experience with the most. Working on issue reports that are novel to them is the least popular objective and selecting issue reports according to the business value falls between the other 2 objectives.

In section 1, developers are also asked to give an estimation of how many issues they could finish in a 2-week sprint. The answers are used to validate the estimated number of completed issues per sprint since the quality of the sprint plan generated relies to some extent on the correctness of the estimation. As a result, one of the answers shows an unrealistically high value (15), while the calculated estimation based on the issue completion history for that developer was significantly smaller (3). Another answer coincided with the estimated value, and the remaining 3 answers were values that differed in 1 issue.

In the last section of the survey, developers are given a set of issues that are generated by the multi-objective approach. Then, developers evaluate the prevalence of the 3 objectives in the set of issues. Figure 11 shows the Likert plot of the answers to these questions. One developer answered that the generated set of 2 issues did not include any business value. The developer explained that the retrieved issues were unimportant bugs: "these are small bugs". The definition of business value is linked to the priority of the issue in this thesis and thus the priority of the issue report can be investigated. The priority of the retrieved issues is "High" and "Low". The low priority issue is marked as a "Bugfix" type and seems to be a small issue, where redirects happen before navigating to

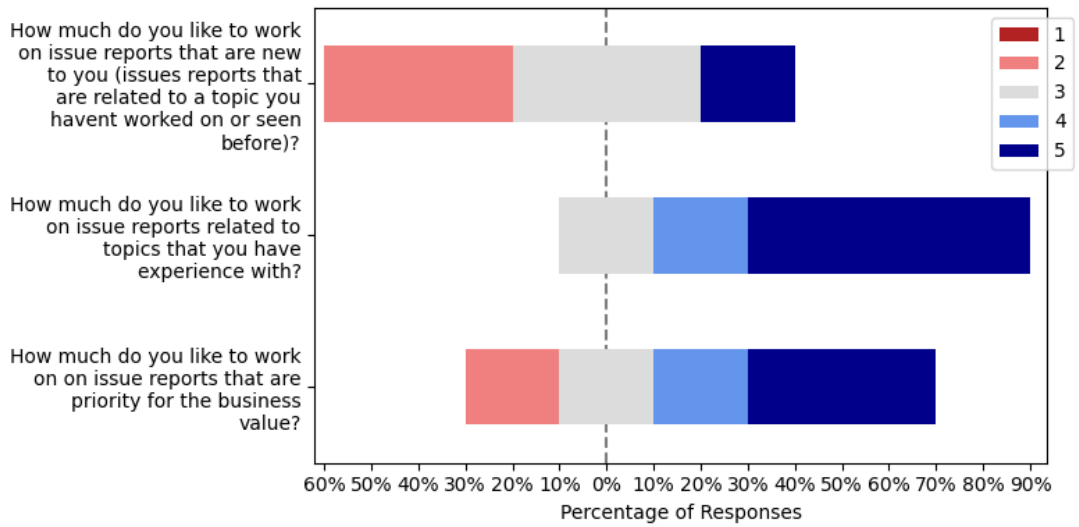

Figure 10. Survey section 1 answers

a page. The issue with high priority is marked with a type "Maintenance" and seems to be of higher impact. From the issue description, it can be understood that some functionality of a payments integration is not working properly in some cases. This could suggest that the issue was incorrectly prioritized or the developer evaluated the business value incorrectly. Also, there could be a mismatch between the developer's idea of priority and the idea of business value. Additionally, at the time of analyzing the answers, the high priority issue was completed by the developer, for whom it was recommended to prove it had business value, and the low priority issue is still in the backlog.

Another developer answered that the set of retrieved issues did not include any topics that he had experience with (1 on the Likert scale) and it included topics that are new to him (5 on the Likert scale). For that developer, 2 issues are in the set found for him. For one issue the tool correctly evaluates the issue as new for the developer and also recognizes that the developer experience objective is very low. The issue is prioritized as "High", which means that it was included in the set because 2 of the objectives had very high values and developer experience didn't matter as much. For the second issue on the other hand the developer experience objective is very high, the priority of the issue is "Low" and the issue is not recognized as being novel. This means that the tool is completely wrong on this issue since this issue is in the set only for the high developer

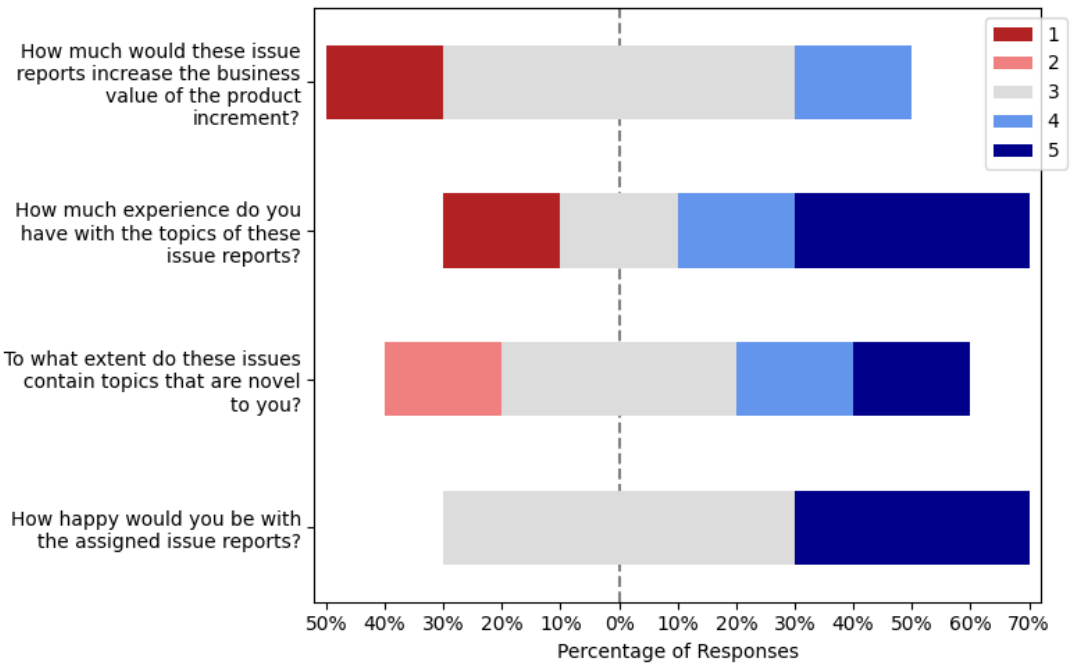

Figure 11. Survey section 3 answers

experience objective. The problem with this issue might be in the issue description. The description includes 2 vague sentences without any details being described and it is probably difficult to understand which topics are included in the issue.

In the second section of the survey, developers manually select issue reports. The fitness function scores are calculated for each objective for the set issues and compared against the fitness function scores of generated set of issues. The picked set of issues represents a good solution from the perspective of the developer and thus the multi-objective approach should rate this set of issues highly. GD is used to measure the distance of the picked set from the Pareto front. A smaller distance indicates a better solution since the Pareto front represents the most optimal solutions. The distances can be seen in Table 6. For developer 1 the hand-picked solution is close to the best solution as can be in Figure 12. Business value and novelty objectives are equal, but the experience objective is slightly lower. GD was second-lowest for this developer which means that the hand-picked solution is also rated highly by the tool. The approach also rates the hand-picked set of issues of developer 2 highly, which is proved by the

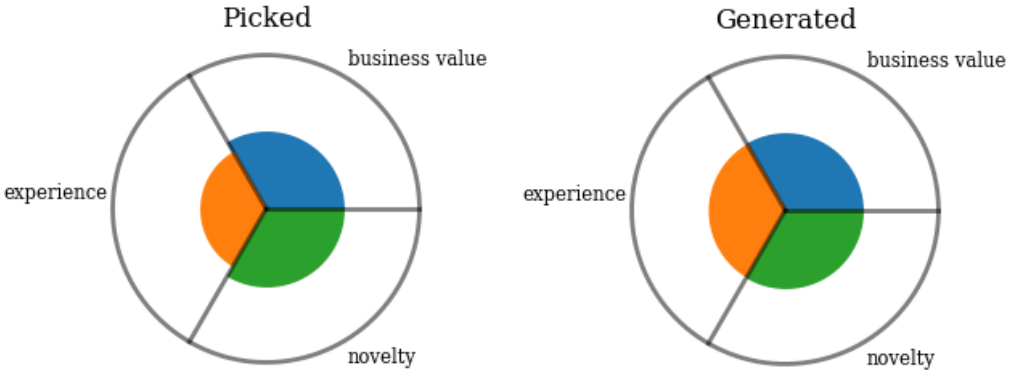

Figure 12. Developer 1 Petal diagram comparing the objective values of the picked solution to the generated solution

lowest GD. For this developer, the distribution of objectives is different between the generated and hand-picked solutions, which can be seen in Figure 13. The generated solution includes a novel issue report, but the experience objective is lower compared to the manually selected set. For developer 3, experience and business value are both worse for the picked solution and neither solution includes any novelty as seen in Figure 14. Figure 15 shows that the picked issues included more business value at the expense of experience for developer 4. The picked solution of developer 5 is rated poorly by the tool since all of the objectives are worse compared to the generated one. GD of the picked solution to the Pareto front is also the largest for this developer. This can be seen in Figure 16. I looked further into the data since the generated sets for developer 3 and 4 are optimized for only business value and experience. I found that the backlog doesn't include any novel issues for these developers. In cases where novel issues are available for developers, the approach optimizes for all 3 objectives.

Additionally, developers are asked how long it took to go over the backlog and manually select the issues for the next sprint. Four out of five developers reported that the task takes a few minutes: "1 minute", "5 minutes", "2 minutes" and "a couple of minutes" were the reported answers. One answer was not considered as it was not realistic (2 seconds). When comparing this time with the execution time of the tool, the automatic tool retrieved the set of issues in over 0.6 seconds out of the backlog containing 18 issues. In addition to the time, they were asked what was the most difficult part when picking

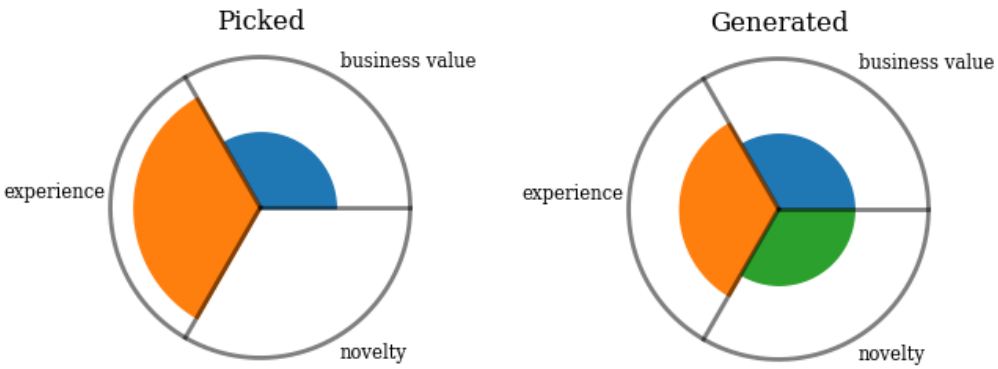

Figure 13. Developer 2 Petal diagram comparing the objective values of the picked solution to the generated solution

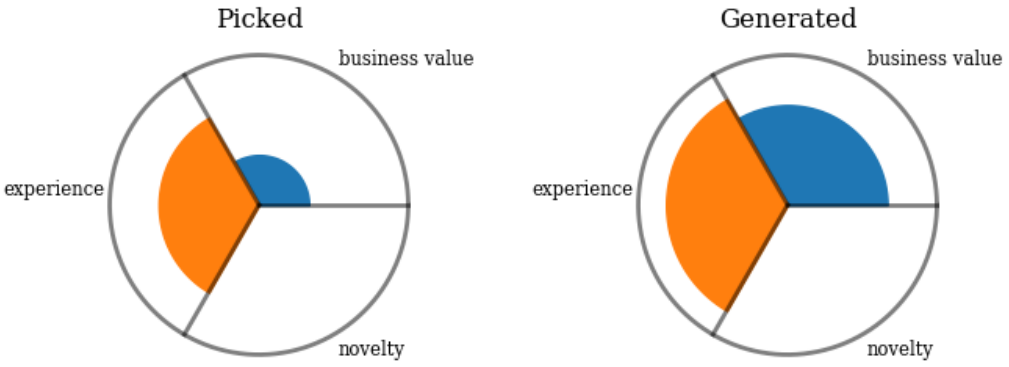

Figure 14. Developer 3 Petal diagram comparing the objective values of the picked solution to the generated solution

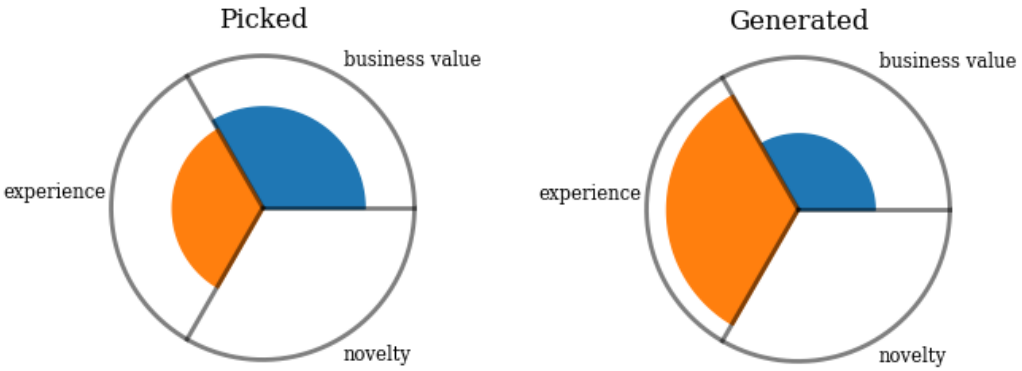

Figure 15. Developer 4 Petal diagram comparing the objective values of the picked solution to the generated solution

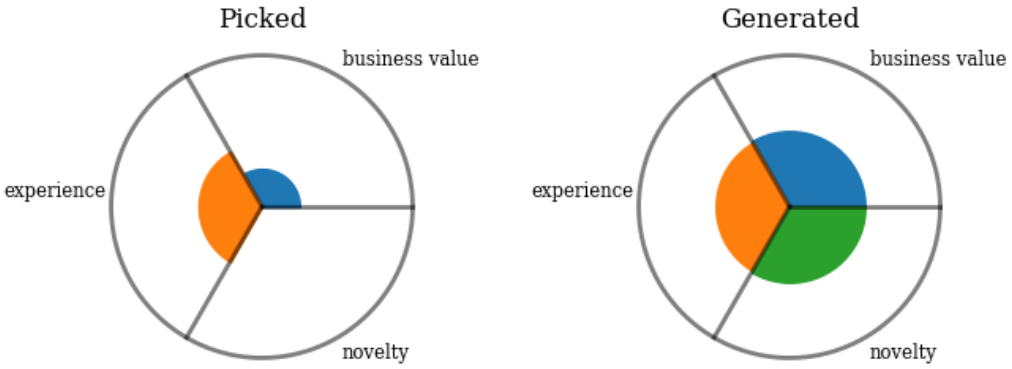

Figure 16. Developer 5 Petal diagram comparing the objective values of the picked solution to the generated solution

the issues. Among the answers were "reading" and "no idea what 99% of these issues entail", confirming that the task is not trivial and requires effort.

An interesting answer was "finding relation to the current OKR", which means that the issues are not always considered in isolation. OKR references a goal-setting framework used by teams and organizations that encourages the definition of measurable goals and tracking of their outcomes. In Bondora, quarterly cycles are used and each quarter new business goals are set. This means that issues planned into sprints should be about helping achieve the goals set that quarter. Similar to Al-Zubaidi et al. [34], an additional objective could be considered in future works, which allows to state the current business goal and find issues that align with that goal.

None of the respondents rated the assigned set of issue reports negatively. 60% of the answers were neutral (3 on the Likert scale) and 40% were rated highly (5 on the Likert scale). The comparison of manually selected issues and the generated set of issues showed that the approach works well in some cases. It optimizes for all 3 objectives in cases where the backlog is rich in terms of the availability of all objectives. A large benefit of the proposed approach is the time performance. In the survey, it took 1-5 minutes for the developers to pick 2-3 issues from the backlog that consisted of 18 issues while it took about half a second for the tool proposed in this thesis. For reference, in the open-source dataset, most projects had more than 100 issues in the backlog, which increases the manual composition of the sprint plan by a lot or makes it infeasible. For that reason, the tool would be beneficial in the sprint planning processes of software projects.

6 Discussion

6.1 Research questions

The LDA execution time results show that projects with a larger number of done issues require more time as seen in Figure 7. It is possible to build the models offline and save them into files for later. In case that new issue reports are added to the project, the model requires an update. LDA does not require re-training the model since it supports the update of the model by simply adding unseen documents to the model. This incremental update presents an advantage of the approach. Similarly, it is also possible to calculate the developer experience and topic distributions of backlog issues offline, which reduces the total execution time required to generate a set of issues for a developer.

NSGA-II optimization time increases as the backlog size increases (Figure 8). A larger backlog means that the search space is larger and the algorithm requires more time to explore that space and find optimal solutions. For this reason, the approach requires more computational power for projects with more than 8000 issue reports in their backlog. In the dataset used, this is the case of the MDL project. Sprint planning usually doesn't take place with a higher frequency than a week and the fact that the average optimization time is less than 8 seconds for projects with under 1000 issues is acceptable from the user experience standpoint.

The survey responses show that developer preferences are in accordance with previous research [16]. Every developer in this survey takes novelty, experience with the topics of the issue, and business value into account. Novelty is the least popular characteristic of the issue followed by the business value of the issue. The most popular property of the issue when self-assigning was the developer's experience with similar issues in the past. The presented approach currently considers these preferences for selecting the set of issues but the optimization algorithm assigns equal importance to the preferences (i.e., the objectives). This configuration doesn't match the preferences shown by the developers who answered the survey. The answers show that the preferences are individual and suggest that different weights for each objective should be taken into account when running the algorithm.

The formula used to quantify the business value might impact on the quality of the solutions. Business value is linked to the priority of the issue report in this thesis.

As one developer indicates, there are cases where the approach considers a small bug important and the developer doesn't, despite the bug being highly prioritized. This indicates that there could be a mismatch between the developer's idea of business value and the definition used in this thesis. There is no agreement about what business value means. As shown by Gregory et al. [9], the idea of business value varies from team to team.

6.2 Limitations

This study presents several limitations that are worth mentioning. First, the limited access to developers willing to participate in the study led to a small sample of survey responses. Although the findings obtained from the survey analysis cannot be generalizable, they are useful to explain the quantitative results and form hypotheses for further research. The respondents were also my colleagues at the time of writing this thesis, which might have included bias in the results.

The quantitative research revealed several edge cases with high HV and a high number of issues. Since the number of developers, who completed the qualitative research is low, there isn't enough data to explain the reasons behind the edge cases or improve the approach and thus more research is needed.

Second, the dataset used in the qualitative research included only 18 backlog issues, which might not be representative of a larger software project. For example, the backlog could randomly include issues with higher business value than a more representative sample would.

The algorithm used for optimization is NSGA-II, which is one of the most popular multi-objective algorithms [31]. Although NSGA-II showed acceptable results, there are several multi-objective algorithms that can be used. Therefore, there might be other optimization algorithms that perform better than NSGA-II.

LDA is used for topic modeling in this thesis. The algorithm is one of the most popular for this purpose in the field of Natural Language Processing [13]. There are many LDA variants and other alternatives, which could be explored to improve the topic modeling performance.

7 Conclusions and future work

Automatic tools for iteration planning are beneficial to software teams using Agile methodology in order to save time. This is especially relevant for projects with large backlogs. Accounting for developers' preferences in addition to the business value aims to increase developers' motivation and productivity when composing the sprints. Taking these statements into account, an automatic approach is proposed in this thesis, that retrieves issues from the backlog for a developer based on their past velocity and optimizes for the business value of the issue, developer's previous experience with similar issues, and the novelty of the issue. The approach uses LDA to extract topics from the issues, which helps to define the developer experience and novelty objectives. NSGA-II is used as the optimization algorithm to extract the best combinations of issues from the backlog.

The thesis set out to answer 2 research questions: what is the performance of the sprint planning approach and what is the utility of the approach as perceived by developers. A performance of 20%-64% from the highest weighted average HV is expected for software projects that want to implement this approach. The best results are found in less than 8 seconds for projects with less than 1000 backlog issues. For projects with larger backlogs, the approach might not be viable, since calculation takes too much time.

Plenty of topics should be considered for future works. NLP models and optimization algorithms should be compared to find out the best methods for generating sprint plans. In addition, a different optimization algorithm might be more suitable for very large backlogs as the proposed approach is not viable for such cases. Also, it would be interesting to use the sprint planning tool on a dataset with complete story points data. This would allow using a business value definition that includes story points and also help to generate solutions that include a more accurate amount of work compared to the amount of work developers are able to complete. Future works should also take into account the differences in developer preferences when choosing issues for a sprint. Practical implementations should allow users to define the weights of the objectives to get more personalized results. The survey conducted in this thesis should also be used on a larger sample of developers in order to thoroughly validate the usefulness of the approach in real-life scenarios. Since the performance was difficult to evaluate based on

the selected experiment and performance metric, other approaches should be considered for future works.

References

[1] Pekka Abrahamsson et al. “Agile Software Development Methods: Review and Analysis”. In: *Proc. Espoo 2002* (Jan. 2002), pp. 3–107.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O’Reilly Media, Inc.", 2009.

[3] J. Blank and K. Deb. “pymoo: Multi-Objective Optimization in Python”. In: *IEEE Access* 8 (2020), pp. 89497–89509.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation”. In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[5] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[6] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. “An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator”. In: *2006 IEEE International Conference on Evolutionary Computation*. 2006, pp. 1157–1163. DOI: 10.1109/CEC.2006.1688440.

[7] Daniel Graziotin et al. “Consequences of Unhappiness While Developing Software”. In: (Jan. 2017).

[8] Daniel Graziotin et al. “Unhappy Developers: Bad for Themselves, Bad for Process, and Bad for Software Product”. In: May 2017, pp. 362–364. DOI: 10.1109/ICSE-C.2017.104.

[9] Peggy Gregory et al. “STAKEHOLDER PERCEPTIONS OF IT BUSINESS VALUE IN A PUBLIC SECTOR IT DIGITALISATION PROJECT”. In: May 2020.

[10] Nyoman Gunantara. “A review of multi-objective optimization: Methods and its applications”. In: *Cogent Engineering* 5.1 (2018). Ed. by Qingsong Ai, p. 1502242. DOI: 10.1080/23311916.2018.1502242. eprint: https://doi.org/10.1080/

23311916.2018.1502242. URL: https://doi.org/10.1080/23311916.2018.1502242.

[11] Hisao Ishibuchi et al. “How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison”. In: *Evolutionary Computation* 26 (May 2018), pp. 1–29. DOI: 10.1162/evco_a_00226.

[12] Hisao Ishibuchi et al. “Modified Distance Calculation in Generational Distance and Inverted Generational Distance”. In: *Evolutionary Multi-Criterion Optimization*. Ed. by António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello. Cham: Springer International Publishing, 2015, pp. 110–125. ISBN: 978-3-319-15892-1.

[13] Hamed Jelodar et al. “Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey”. In: *Multimedia Tools and Applications* 78.11 (June 2019), pp. 15169–15211. ISSN: 1573-7721. DOI: 10.1007/s11042-018-6894-4. URL: https://doi.org/10.1007/s11042-018-6894-4.

[14] Bart Luijten, Joost Visser, and Andy Zaidman. “Assessment of issue handling efficiency”. In: *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. 2010, pp. 94–97. DOI: 10.1109/MSR.2010.5463292.

[15] Zainab Masood, Rashina Hoda, and Kelly Blincoe. “How agile teams make self-assignment work: a grounded theory study”. In: *Empirical Software Engineering* 25 (Nov. 2020), pp. 1–44. DOI: 10.1007/s10664-020-09876-x.

[16] Zainab Masood, Rashina Hoda, and Kelly Blincoe. “Motivation for Self-Assignment: Factors Agile Software Developers Consider”. In: *2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 2017, pp. 92–93. DOI: 10.1109/CHASE.2017.18.

[17] Gurpreet Matharu et al. “Empirical Study of Agile Software Development Methodologies”. In: *ACM SIGSOFT Software Engineering Notes* 40 (Feb. 2015), pp. 1–6. DOI: 10.1145/2693208.2693233.

[18] Yury Nikulin, Kaisa Miettinen, and Marko Mäkelä. “A new achievement scalarizing function based on parameterization in multiobjective optimization”. In: *Operations Research-Spektrum* 34 (Aug. 2012), pp. 69–87. DOI: 10.1007/s00291-010-0224-1.

[19] Samir Omanovic and Emir Buza. “Importance of stable velocity in agile maintenance”. In: *2013 XXIV International Conference on Information, Communication and Automation Technologies (ICAT)*. 2013, pp. 1–8. DOI: 10.1109/ICAT.2013.6684044.

[20] Radim Rehurek and Petr Sojka. “Gensim–python framework for vector space modelling”. In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011).

[21] Marco Tulio Ribeiro et al. “Multiobjective Pareto-Efficient Approaches for Recommender Systems”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5 (2014), pp. 1–20.

[22] Ken Schwaber and Jeff Sutherland. “Der Scrum Guide”. In: (2014).

[23] Ezequiel Scott, Khaled Nimr Charkie, and Dietmar Pfahl. “Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects”. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2020, pp. 124–131. DOI: 10.1109/SEAA51224.2020.00029.

[24] Ezequiel Scott and Dietmar Pfahl. “Using Developers’ Features to Estimate Story Points”. In: *Proceedings of the 2018 International Conference on Software and System Process*. ICSSP ’18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 106–110. ISBN: 9781450364591. DOI: 10.1145/3202710.3203160. URL: https://doi.org/10.1145/3202710.3203160.

[25] Ezequiel Scott, Tanel Tõemets, and Dietmar Pfahl. “An Empirical Study of User Story Quality and Its Impact on Open Source Project Performance”. In: *Software Quality: Future Perspectives on Software Engineering Quality*. Ed. by Dietmar Winkler et al. Cham: Springer International Publishing, 2021, pp. 119–138. ISBN: 978-3-030-65854-0.

[26] Ke Shang et al. “A Survey on the Hypervolume Indicator in Evolutionary Multiobjective Optimization”. In: *IEEE Transactions on Evolutionary Computation* 25.1 (2021), pp. 1–20. DOI: 10.1109/TEVC.2020.3013290.

[27] Marian Stoica, Marinela Mircea, and Bogdan Ghilic-Micu. “Software Development: Agile vs. Traditional”. In: *Informatica Economica* 17 (Dec. 2013), pp. 64–76. DOI: 10.12948/issn14531305/17.4.2013.06.

[28] Shaheen Syed and Marco Spruit. “Selecting Priors for Latent Dirichlet Allocation”. In: *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. 2018, pp. 194–202. DOI: 10.1109/ICSC.2018.00035.

[29] Suppawong Tuarob et al. “Automatic team recommendation for collaborative software development”. In: *Empirical Software Engineering* 26.4 (May 2021), p. 64. ISSN: 1573-7616. DOI: 10.1007/s10664-021-09966-4. URL: https://doi.org/10.1007/s10664-021-09966-4.

[30] Theresa Velden et al. “Comparison of topic extraction approaches and their results”. In: *Scientometrics* 111.2 (May 2017), pp. 1169–1221. ISSN: 1588-2861. DOI: 10.1007/s11192-017-2306-1. URL: https://doi.org/10.1007/s11192-017-2306-1.

[31] Shanu Verma, Millie Pant, and Vaclav Snasel. “A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems”. In: *IEEE Access* 9 (2021), pp. 57757–57791. DOI: 10.1109/ACCESS.2021.3070634.

[32] Darrell Whitley. “A genetic algorithm tutorial”. In: *Statistics and Computing* 4.2 (June 1994), pp. 65–85. ISSN: 1573-1375. DOI: 10.1007/BF00175354. URL: https://doi.org/10.1007/BF00175354.

[33] Aimin Zhou et al. “Multiobjective evolutionary algorithms: A survey of the state of the art”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2011.03.001. URL: https://www.sciencedirect.com/science/article/pii/S2210650211000058.

[34] Wisam Haitham Abbood Al-Zubaidi et al. “Multi-Objective Iteration Planning in Agile Development”. In: *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. 2018, pp. 484–493. DOI: 10.1109/APSEC.2018.00063.

Appendix

I. Interview questionnaire

7.0.1 Survey introduction

The topic of the thesis is proposing an automatic sprint planning helper that retrieves a personalized set of issues for a developer for the upcoming sprint. With this survey, I ask you to provide information about your preferences when selecting issues and evaluate the personalized set of issues retrieved by the automatic tool.

All the information you provide in this survey is strictly treated as confidential, that is, your name, the issue reports, or any other sensitive data will not be disclosed. Your answers will be used only for academic purposes and only aggregated data will be published in the thesis report. This survey is unrelated to Bondora or any other commercial company.

7.0.2 Section 1: Demographic

Questions:

1.
   - Question: What is your name?
   - Type: Short text
   - Required: Yes
2.
   - Question: What is your job title?
   - Type: Short text
   - Required: Yes
3.
   - Question: What are your responsibilities?
   - Type: Short text
   - Required: Yes
4.
   - Question: How long have you been in the role at your current company?
   - Type: Short text

- Required: Yes

5. - Question: How much do you like to work on issue reports that are new to you (issues reports that are related to a topic you haven't worked on or seen before)?
   - Type: Likert scale, 1-5
   - Required: Yes

6. - Question: How much do you like to work on issue reports related to topics that you have experience with?
   - Type: Likert scale, 1-5
   - Required: Yes

7. - Question: How much do you like to work on issue reports that are a priority for the business value.
   - Type: Likert scale, 1-5
   - Required: Yes

8. - Question: How many issue reports do you usually complete in a 2-week sprint?
   - Type: Short text
   - Required: Yes

7.0.3 Section 2: Choosing the desired sprint plan

Intro to section 2:

Now, please complete the following task: The following list of issue reports is taken from a recent backlog of the project. Please select a set of issue reports that you would like to work on in during the next sprint.

When selecting the issue reports, please take into account the following points:

Questions:

9. - Question: Select 2 issue reports from the following backlog:

- Type: Checkboxes
- Options: Backlog issues (summary and link to issue tracking system)
- Required: Yes
- Validation: Selected number of issues have to match the developer's calculated velocity

10.
- Question: Select 2 issue reports from the following backlog:
- Type: Checkboxes
- Options: Backlog issues (summary and link to issue tracking system)
- Required: Yes
- Validation: Selected number of issues have to match the developer's calculated velocity

11.
- Question: How long did it take to select these issues?
- Type: Short text
- Required: Yes

12.
- Question: What was the most difficult part?
- Type: Short text
- Required: Yes

13.
- Question: If you had problems with selecting the issues, explain them here.
- Type: Short text
- Required: No

7.0.4 Section 3: Evaluating the generated set of issues

Intro to section 3: The list below contains a set of issue reports that were automatically selected from your team backlog according to your profile.

[Set of issues retrieved for the developer]

If these issue reports were assigned to you to be completed in the next sprint, ...

14.
- Question: Would it be feasible to complete these issue reports in the next sprint?
- Type: Multiple choice
- Options:
  - Yes
  - No
- Required: Yes

15.
- Question: The amount of work is...
- Type: Multiple choice
- Options:
  - Underestimated - too little work for 2 weeks
  - Correctly estimated
  - Overestimated - the amount of work cannot be completed in 2 weeks
- Required: Yes

16.
- Question: How much would these issue reports increase the business value of the product increment?
- Type: Likert scale, 1-5, 1-Not much, 5-A lot
- Required: Yes

17.
- Question: Please explain the answer in more detail.
- Type: Long text
- Required: Yes

18.
- Question: How much experience do you have with the topics of these issue reports?
- Type: Likert scale, 1-5, 1-I don't know the topics, 5-I know the topics
- Required: Yes

19.
- Question: Please explain the answer in more detail.

- Type: Long text
- Required: Yes

20.
- Question: To what extent do these issues contain topics that are novel to you?
- Type: Likert scale, 1-5, 1-The issues are known, 5-The issues are unknown
- Required: Yes

21.
- Question: Please explain the answer in more detail.
- Type: Long text
- Required: Yes

22.
- Question: How happy would you be with the assigned issue reports?
- Type: Likert scale, 1-5, 1-Unhappy, 5-Happy
- Required: Yes

23.
- Question: Please explain the answer in more detail.
- Type: Long text
- Required: Yes

24.
- Question: Any further comments?
- Type: Long text
- Required: No

II. Prototype

A prototype web app was built, where the approach can be tested using the datasets used in the quantitative experiment. The prototype web app uses Flask[3] backend and Angular[4] frontend. Figure 17 shows the dataset selection view. After selecting the desired dataset, the user is redirected to the optimization page, shown in Figure 18. The developer drop-down includes a list of all developers in the selected dataset. After selecting the developer, the number of issues that represent the work capacity is calculated for the developer. The input is automatically filled with the computed number. Submitting the form generates the optimal set of issues and displays values of all objectives in a radar diagram.

[3]https://flask.palletsprojects.com
[4]https://angular.io

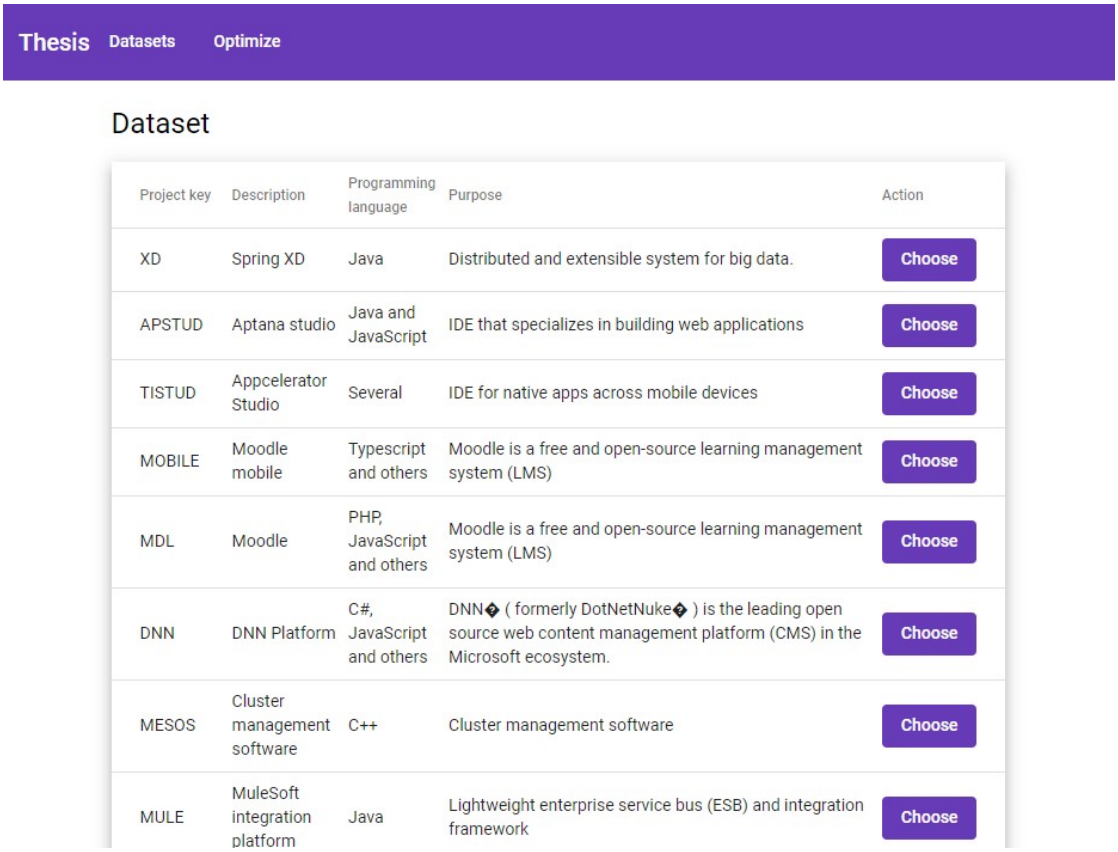

Figure 17. Dataset selection view

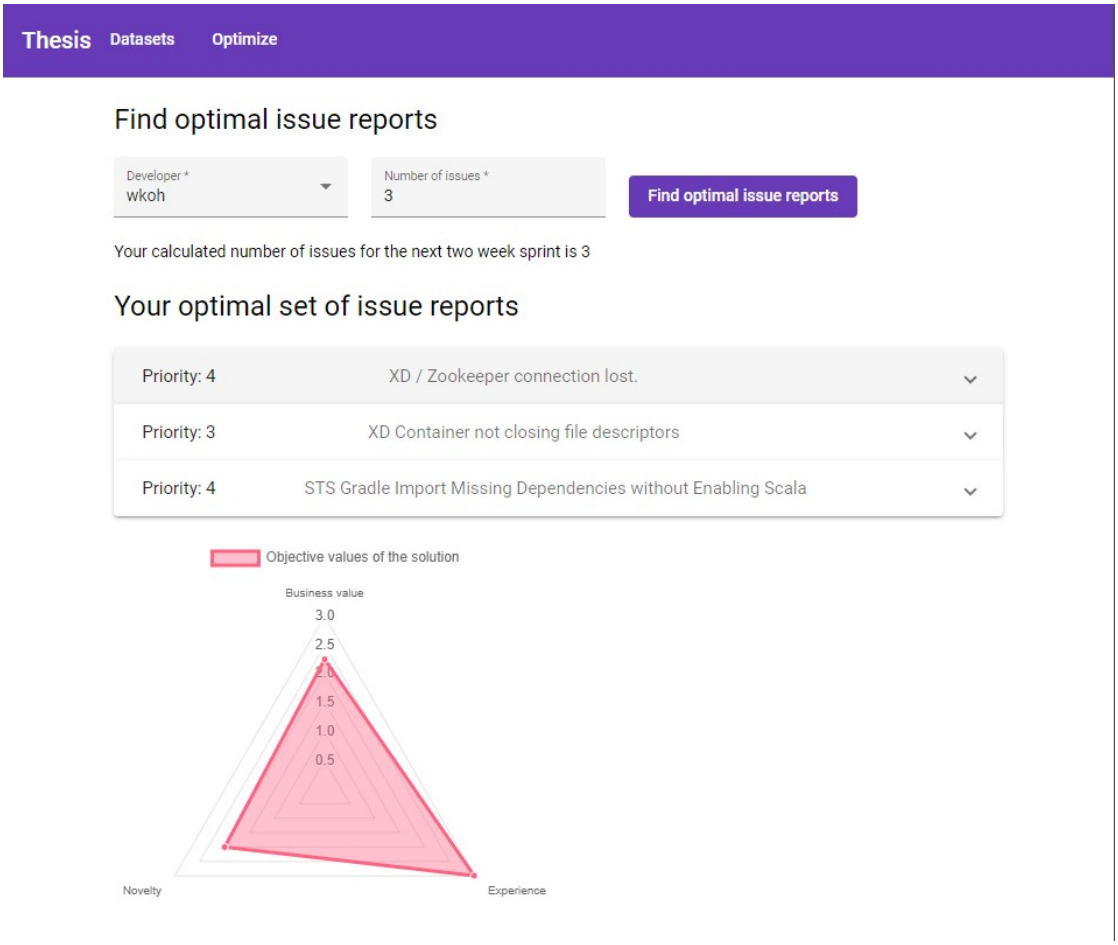

Figure 18. Optimization view with the results and a radar diagram depicting the values for each objective.

III. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Sander Jenk**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   A multi-objective optimizer to retrieve issue reports based on developer experience and business value,
   *(*title of thesis)

   supervised by Ezequiel Scott, PhD.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Sander Jenk
17.05.2022