Evolutionary Robust Optimization for Software Product Line Scoping: An Explorative Study

Reza Karimpour\textsuperscript{1,*}, Guenther Ruhe\textsuperscript{2}

\textit{Software Engineering Decision Support Lab}

\textit{Computer Science Department, University of Calgary}

\textbf{Abstract}

\textbf{Background:} Software product line (SPL) scoping is an important phase when planning for product line adoption. A SPL scope specifies: (1) the extend of domain supported by the product line, (2) portfolio of products in the product line and (3) list of assets to be developed for reuse across the family of products.

\textbf{Issue:} SPL scope planning is usually based on estimates about the state of the market and the engineering capabilities of the development team. One challenge with these estimates is that there may be inaccuracies due to uncertainty in the environment or accuracy of measurement of estimates. This may result in issues ranging from suboptimal plans to infeasible plans.

\textbf{Objective:} To address the issue above, we propose to include uncertainty as part of the SPL scoping model. Plans developed in consideration of uncertainty would be more robust against possible fluctuations in estimates.

\textbf{Approach:} In this paper, a method to incorporate uncertainty in scoping optimization and its application to generate robust solutions is proposed. We capture uncertainty as part of the formulation and model scoping optimization as a \textit{multi-objective} problem with \textit{profit} and \textit{stability} as fitness functions. Profit stability and feasibility stability are considered to represent stability concerns. To measure the effectiveness of our proposal and evaluate its validity, five different experiments are conducted.

\textbf{Results:} Results show that compared to other scope optimization approaches, both performance stability and feasibility stability are improved while maintaining near optimal performance for profit objective. Also, generated results comprise of solutions ranging a variety of trade-offs between profit and stability, providing the decision maker with enhanced decision support.

\textbf{Conclusion:} Multi-objective optimization with stability consideration for SPL scoping provides project managers both a robust and flexible way to address uncertainty in the process of SPL scoping.

\textit{Keywords:} Search-based software engineering, Software product line portfolio

\*Corresponding author

\textsuperscript{1}reza.karimpour@ucalgary.ca

\textsuperscript{2}ruhe@ucalgary.ca

Preprint submitted to Computer Languages, Systems & Structures

April 12, 2016
Software product line (SPL) is a development approach built on the idea that similar systems, which vary in some details, can be developed out of a single architecture. The benefits of this approach are observable in cost cuts and market agility [1]. Since the adoption of SPL requires significant upfront investment [2], it is plausible to validate the economical justification and profitability of such decision. Achieving this requires clear understanding of the domain and scope of product line.

**SPL Scoping** is an activity concentrated on specifying different scope related decisions including Asset Scoping, Domain Scoping and Portfolio Scoping. Asset scoping includes decisions about which assets of a product line are to be developed for reuse across different products. Portfolio Scoping includes decision about which products to be included as part of the product line portfolio. The final category of decisions in SPL Scoping is Domain Scoping that includes decisions about the extend of the domain supported by the product line [3].

An example of scoping is given in Figure 1. This software product line consists of three candidate products, two customer segments and two competitors. As demonstrated, the product High-end is allocated to Enterprise segment and priced as 280. Starter product is allocated to the Small business segment and Expert product is not selected for production. As for asset scoping, Accounting feature is planned to be built for reuse while Sales and Web-interface are custom developed for each product. Marketing feature is skipped since it is exclusive to the Expert product. In summary, in this study, the decision of which product to build, what product to offer to a segment, and how to price a product comprise the scoping plan.

![Figure 1: A scoping example](image-url)
As reported in industrial case studies, among the three scoping types, product scoping and asset scoping require more effort and resources [4]. Also, asset scoping and product scoping greatly influence development activities. The two major development activities, Asset development and Product development [5], are affected by the scope defined in pertaining scoping activities.

Current methods that model SPL scoping are mainly based on ROI [6]. In these approaches, profit is the main directing measure and is defined as a function of cost and revenue: \( \text{profit} = \text{revenue} - \text{cost} \) [7]. For example, Müller [7] model scoping as a profit maximization problem and Gillain et al. [8] developed goal based requirement modelling and NPV based valuation.

1.1. Motivation

1.1.1. Robust SPL scoping

Although, once adopted, a SPL may help reducing risks associated with software production [3, 9], yet the decision of whether to produce a specific product or not and if a specific asset should be developed in a reusable way, requires careful evaluation. One main reason for this is that most scoping approaches depend on some input data that are generally estimated. This input data includes cost estimates and market assessment to name but two. Depending on the type of the project, this estimation may come from stakeholders, marketing, development team, or management [10]. Most research papers do not address inaccuracies and uncertainties in estimates [10, 11].

Many software engineering tasks rely on estimates that may be inaccurate, insufficient and are likely to change [12]. In case of SPL scoping, estimated measures are, among others, willingness-to-pay, production cost, and competitors. Willingness-to-pay stands for the cost a customer is willing to pay for a product. On the other hand, production cost depicts the effort required to produce a product. Finally, estimates about competitors include their offerings and marketing strategies. Being estimates mean that these quantities may change as they are captured from environment and since the environment is not static, these numbers can also change over time. There is also the risk of error in the estimation process. For example, one may under-estimate the production cost of an asset while another may over-estimate willingness-to-pay of a customer segment.

1.1.2. Trade-off decision making

SPL managers need to make early decision in the lifecycle of a product line. Therefore, they need to consider the possibility of unexpected changes to variety of variables. This often requires making trade-off decisions between choosing a very strict but fragile plan or deviate from optimality to gain robustness and stability.

Multi-objective evolutionary algorithms that generate a non-dominated front [13, 14] provide convenient support for trade-off analysis. They provide decision makers with different options ranging from one end of spectrum to the other.
1.2. Hypothesis and approach

Recently, search-based approaches to software product line engineering problems have become popular [15], with scoping not receiving much attention. In this study, we propose a method based on evolutionary optimization that performs SPL scoping while considering uncertainty in estimates. We adopt the scoping model from Müller [7] and expand it by adding support for robustness. This is achieved by adding extra objectives that reflect stability and measure the quality of solution with regard to uncertainty. Then, we evaluate the effectiveness and performance of our proposal using simulation and compare it with other approaches. For implementation, we offer new crossover and mutation operators specially designed for this problem to achieve reasonable performance.

1.3. Goal statement

The goal of this study is to propose and evaluate an approach for considering estimation inaccuracies and environment uncertainties in SPL scoping optimization. The main research goal of this report is to evaluate how SPL scoping could benefit from robust evolutionary optimization. This general research goal is represented by four research questions in Section 5.

1.4. Novelties and contributions

The main contribution of this paper is the addition of the concept of stability in SPL scoping and showing how multi-objective evolutionary algorithms can help performing trade-off analysis of profit and stability for this software engineering problem.

The main contributions of this paper are:

- **Introduction of robust optimization in SPL scoping.** We propose and evaluate a multi-objective model to target scoping designs that have higher stability while still maintaining near-optimal performance (profit).

- **Comprehensive evaluation and comparison of different methods for robust scoping optimization.** In addition to multi-objective approach, we also implemented and evaluated other scoping methods and performed a comprehensive evaluation and comparison of the results.

- **Comprehensive evaluation and comparison of different evolutionary algorithms and approaches.** To better evaluate the model, we evaluated our model using three multi-objective evolutionary algorithms, namely NSGA-II [13], SPEA2 [14], and MOCell [16].

- **Novel mutation and crossover operators.** We designed and evaluated a set of operators that yield better results compared to general-purpose operators.

Initial results of this study were published in Karimpour and Ruhe [17]. The current paper extends the previous work with additional evaluations and experiments. These additions include:
• Comprehensive report on related background literature and study design decisions
• Formal SPL scoping model and the uncertainty model used in this study
• Addition of RSO as the third evaluated method
• Comprehensive report on the evolutionary operators and the encoding used for implementation
• Additional evaluations showing the effect of different sampling sizes, different sampling methods and different evolutionary algorithms
• Comprehensive discussion of threats to validity and mitigation strategies taken in this study

1.5. Structure of this paper

In Section 2, a brief overview of related work is given. Section 3 defines robust scoping. Next, Section 4 lays out different approaches evaluated in this study. Section 5 lists research questions and explains details about the design and implementation. In Section 6, we report and discuss obtained results. Finally, Section 7 concludes this paper.

2. Background and related work

2.1. SPL scoping

SPL engineering has two main stages. The initial stage is Domain-engineering. The purpose of this stage is to capture and model the characteristics of the domain for which the SPL will be developed. In domain-engineering, possible products and their shared and unique features are determined. Application-engineering is the next stage in which new applications based on the details of a specific instance of the domain are developed [18]. In other words, the domain-engineering phase finds and documents the variability points while the application-engineering binds those variability points to concrete instances. SPL Scoping mostly falls under domain-engineering.

Schmid [19] proposed the PuLSE-Eco V2.0 scoping guideline with three main steps as depicted in Figure 2. This guideline requires portfolio of existing systems and potential products. The first step takes this data and performs product line mapping by generating an abstract description of the SPL along with a feature model. The next step performs assessment on risks and benefits of reuse. Finally, the last step decides about how assets are developed.

![Figure 2: PuLSE-Eco framework [19]](image-url)
Our optimization approach supports all steps of this guideline. For product line mapping, we provide recommendation about which products and assets to be included in the product line plan. As for the assessment phase and asset scoping, we consider different options for implementing assets and provide analysis of how different decisions may affect the robustness of final product line.

**SPL adoption and upfront investment.** There are two different approaches toward SPL adoption with regard to how early SPL structure is planned:

1. In the first approach, also known as greenfield, the whole SPL is planned in advanced and it basically starts from scratch without reusing existing code or other assets. This requires investing a substantial effort in scoping stages including domain scoping and asset scoping. Consequently, break-even point in ROI is postponed increasing the risk of investment.

2. The second approach, also known as brownfield, relies on adopting already existing assets and incremental adoption of SPL. In this case existing software artifacts are used for building reusable components. Therefore, upfront investment is lower and return on investment may be achieved earlier.

As the second approach is mainly based on short iterations and feedbacks, planning for far future is not recommended nor practiced as heavily as the first one. In this research we concentrate on the first type of SPL adoption in which a lot of upfront investment is put into scoping, though our approach could be used for existing systems as well.

**2.2. Uncertainty in estimates**

To address this issue of inaccuracy in estimates for SPL planning, Cantor [10] acknowledged the uncertainty in different aspects in software projects, like cost and value, and proposed Investment value (IV) for more accurate ROI calculation. IV combines cost, value and risk. Two different formulations for ROI are provided. One that looks at the history of spending and creates values and the other that provides estimates for future projected revenue. The result of the former is a number that represents the value acquired up to today while the latter is a random variable that models the future benefits. One key component of this approach is the application of Monte Carlo technique to find IV out of random variables and distributions.

In our work, we adopted some ideas from this work: for example, the idea of assigning three different levels of uncertainty to the environment. We also use random sampling and Monte Carlo simulation for evaluation of risk. One difference between current work and Cantor [10] is the application of multi-objective optimization for trade-off analysis.

**2.3. Robust optimization**

Bader and Zitzler [20] summarized robust optimization approaches into three groups:
1. In first group, the actual objective function is modified so that in addition to performance, it also represents uncertainty. This is done, for example, by aggregating objective function and robustness measures.

2. The second group resort to additional objectives to handle robustness of optimization results. In these approaches, dedicated objective functions are used to capture risk of a solution.

3. Finally, there are approaches than model robustness as a constraint. In these approaches, the amount of risk pertaining to a solution is limited to a specific range using constraints.

In our work, out of the three, we chose to model uncertainty as additional objectives because of the flexibility and support that this approach brings for a product line manager.

*Types of uncertainty in optimization.* With regard to types of uncertainty in optimization, there are a number of different categorizations. In Beyer and Sendhoff [21] different types of uncertainty for design-optimizations were discussed. These include volatility in environment, attainable level of precision in production, uncertainties in problem modelling, and uncertainties in constraints. Volatility in environment points out the changes that may happen to the operating environment. For SPL scoping, this can affect the validity of estimated input parameters.

More specifically, a categorization of uncertainty in evolutionary optimization was summarized by Jin and Branke [22] into four groups: Noise, environment and realization limits, fitness approximation, and time dependent fitness functions. Noise and fitness approximation can directly affect the computed value of a fitness function. On the other hand, changes in environment can modify the assumptions on which the optimization was based. Finally, there could be limitations in realizing a solution and therefore changes to the decision variables of an optimal solution may be necessary.

As explained before, since scoping-plans heavily depend on estimates, it significantly influences the optimization results. Hence, in this paper, we will focus on uncertainty in environment and try to find scoping plans that are resistant to changes in assumptions/estimates. In addition, to represent uncertainty, we will be using probabilistic measures to model the uncertain parts of the model.

3. **Software product line scoping**

3.1. **Scoping formal model**

Although solutions in search-based approaches are more robust [12], they do not specifically target robustness. A solution is considered *more robust* if the performance/quality/fitness of the solution is *less sensitive* to changes in decision or environment variables compared to another solution [23]. Let’s explore this concept by first showing a simple optimization problem:

An optimization problem could be represented using an objective function $f$, a series of decision variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, a number of environment
variables $\mathbf{a} = \{a_1, a_2, \ldots, a_m\}$ and a subset of $x^n$ called $R$ ($R \subset x^n$), where all constraints are satisfied (feasible solutions) [23]. As depicted in (1), we are interested in finding a decision vector that maximizes (or minimizes) $f$ and also satisfies the constraints.

$$
\begin{align*}
\text{Maximize} & \quad F = f(\mathbf{x}, \mathbf{a}) \\
\text{Subject to} & \quad \mathbf{x} \in \mathbf{R}
\end{align*}
$$

To see how changes in environment variables can affect the fitness of a solution, consider Figure 3a. There are two local maxima $x_1$ and $x_2$. Comparatively, $x_2$ dominates $x_1$ because $f(x_1, a_0) < f(x_2, a_0)$, $a_0$ representing the environment. But the neighbourhood of $x_2$ seems to degrade faster than $x_1$. In other words, moving from $a_0$ to $a_1$, $x_1$ is less sensitive to changes in environment and therefore more stable in terms of fitness performance. Similarly, as depicted in Figure 3b, it is less likely for $x_1$ to become infeasible compared to $x_2$. In other words, $x_1$ has higher feasibility stability.

3.2. Model definition

3.2.1. Value-based scoping

The optimization model provided by Müller [7] is a single objective optimization model that concentrates on the behaviour of customers. It considers concepts like a customer’s willingness-to-pay for a specific product, the price of the products offered by competitors and the cost of building a product. The outputs of the optimization are allocation of products to customer segments and product pricing.

We have made some changes to Müller’s model for this study. To target stability, we model scoping as a multi-objective optimization problem that tries
to create stable solutions considering fluctuations of input parameters. Two main types of risk are considered: profit stability and feasibility stability. We assume that the cost of deployment is negligible. This assumption is reasonable as the new distribution methods and automated deployment techniques have significantly reduced the cost of deployment. Additionally, a simpler model would make evaluation experiments more accurate as it reduces the variables involved. Also, we see asset scoping as an integral part of whole scoping picture and include it in the modelling.

First objective function represents profit. In (2), $f_a$ represents the profit of product line, which is equal to the difference between revenue and cost. $allocated$ is a set of segments to which a product is assigned and $price_{alloc_s}$ is the suggested price of product $p$. $N_s$ is the size of segment $s$ and $alloc_{s,p}$ depicts the allocation of a particular product to a specific customer segment.

$$f_a = \sum_{s \in allocated} (N_s \cdot price_{alloc_s}) - COST$$  \hspace{1cm} (2)

In the above $COST$ is defined as:

$$COST = \sum_{f \in r} COST^c_{COR} + \sum_{p \in build} \sum_{g \in r_p} COST^r_g + \sum_{g \in custom_p} COST^c_g$$  \hspace{1cm} (3)

We assume feature is a type of asset that is directly related to the functionality of a product. In (3) $r$ is the set of reusable features, $r_p$ is the set of features in product $p$ where $r_p \subset r$ and $build$ is the set of all products that have been decided to be built. $custom_p$ is the set of features that are custom developed for product $p$. $COST^c_{COR}$, $COST^r_{f}$ and $COST^c_{f}$ represent the cost of developing a feature in a reusable way, cost of reusing a reusable feature and cost of custom developing a feature. More elaborate discussion on calculating cost of product lines could be found in Böckle et al. [6].

Note that in this model, the interest of customers is modelled via willingness-to-pay. In other activities of product line development, for example requirement prioritization or release planning, interests are often expressed at feature level [24], requirement level [25] or goal level [8].

3.2.2. Constraints

Features constraints. These constraints make sure that if a product is allocated to a segment, then all the features required for that product are built. We also need to avoid building unnecessary features that are not going to be used in any product. In (4) $f_p$ is the set of all features required by product $p$. In other words, $f_p = r_p \cup custom_p$. In this constraint, $fb_f$ is the feature-build decision variable for feature $f$. Valid values for $fb$ are explained in Table 1.

$$\prod_{f \in f_p} fb_f > 0, \forall p \in build$$  \hspace{1cm} (4)

$$\forall f \in \{i|fb_i \neq 0\}, \exists p \in build \Rightarrow f \in f_p$$  \hspace{1cm} (5)
### Decision variables

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{bf}$</td>
<td>How feature $f$ is developed. Valid values are ${0, 1, 2}$ that correspond to not developed, custom developed and developed for reuse.</td>
</tr>
<tr>
<td>$price_p$</td>
<td>The price of product $p$</td>
</tr>
<tr>
<td>$alloc_s$</td>
<td>The product index allocated to segment $s$</td>
</tr>
</tbody>
</table>

### Indices

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>Index of customer segments $s \in [0, m)$</td>
</tr>
<tr>
<td>$p$</td>
<td>Index of products $p \in [0, n)$</td>
</tr>
<tr>
<td>$f$</td>
<td>Index of features $f \in [0, l)$</td>
</tr>
</tbody>
</table>

### Estimates

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WTP_{s,p}$</td>
<td>The willingness-to-pay of customer $s$ toward product $p$</td>
</tr>
<tr>
<td>$COST_{f, R}^f$</td>
<td>The cost of developing reusable feature $f$</td>
</tr>
<tr>
<td>$COST_{f, C}^f$</td>
<td>The cost of custom developing feature $f$</td>
</tr>
<tr>
<td>$COMP_p$</td>
<td>The lowest price of offerings from competitors with similar features to product $p$</td>
</tr>
</tbody>
</table>

Table 1: Summary of decision variables, indices and estimates. We assume that all estimates are uncertain data.

In (4), we check that if all features of every allocated products have a value other than zero or not. In (5) we check if a feature that has an $f_{bf}$ value other than 0, then there should be at least one product that is allocated to a segment.

**Customer behaviour.** There are two constraints to model the behaviour of customers. First, (6) represents customers who prefer to purchase the single product that maximizes the difference between what they are willing to pay and the price of that product. $WTP_{s,p}$ stands for willingness-to-pay of customer segment $s$ toward product $p$.

$$WTP_{s,p} - price_p \leq WTP_{s,alloc_s} - price_{alloc_s}, \quad \forall s \in allocated, p \in build$$  \hspace{1cm} (6)

Second, (7) models how customers make selection among all available products including those from the competitors. Here $COMP_p$ refers to the best offering from competitors for the product $p$. This constraint guides the optimization toward solutions that have lower pricing compared to the pricing of competitor products.

$$WTP_{s,p} - COMP_p \leq WTP_{s,alloc_s} - price_{alloc_s}, \quad \forall s \in allocated, p \in build$$  \hspace{1cm} (7)

Table 1 summarizes the components of the model presented above.

#### 3.2.3 Complexity

In order to proceed with an appropriate optimization approach for the model above, first we need to understand its complexity. Other studies have shown that scoping or its related activities are NP-hard. For example, White et al. [26] showed that deciding which features to include in a product (feature selection) is NP-Hard. Also Kohli and Krishnamurti [27] showed that, from a marketing perspective, product design with the consideration of customer preferences and possible product features is NP-Hard. This makes search-based approaches a promising option due to their inherent support for scalability.
3.2.4. Uncertainty model

In this work, scope of uncertainty is the changes in estimates. In this context, change can happen in one of the following scenarios:

- A new data item is added or removed to/from model. This may include a customer, feature or competitor.
- The value of already existing item is changed. For example, the willingness-to-pay of a segment for a product may change.

When a new data item is added or removed, it may have a minor or significant effect on the model. If the change is addition of a new feature with a low cost, then structure of solutions may not be affected significantly. On the other hand, when a new feature with high cost is added, or a new customer segment is added, the structure of candidate solutions may be affected significantly. For the former case, the changes to the model have similar effect when already existing values are changed. We believe our approach is applicable for the cases where change in structure is not significant or already existing data items are changed. For the latter case, due to the significance of the changes, it may be preferable to rerun the optimization to accommodate the change.

4. Approach and methodology

4.1. Motivation

Recent trends in Search-Based Software Engineering have led to growing interest in applying evolutionary based algorithms in solving software engineering challenges [28]. In addition to being able to handle complex problems with a rather simple approach [29], they could also result in more robust solutions. This is because they could target suboptimal solutions that are less susceptible to change [12].

Motivated by the potential of such algorithms, in this section we explain three approaches toward SPL scoping optimization.

4.2. Robust scoping

To include the uncertainty in the profit driven model presented previously, we considered two general approaches:

1. Model uncertainty as a constraint
2. Model uncertainty as additional objectives

Objectives act like soft-constraints where we would like to have lower or higher characteristics in solutions where as constraints impose hard conditions on the final solution. In other words, when a constraint is not satisfied the whole solution is deemed infeasible and possibly discarded while objective functions show how far a solution is from a desired point in the solution space. We see robustness to be sharing more characteristics with soft-constraints where it is the product line manager that decides about the overall acceptance of the robustness level rather than being decided by an automated algorithm. Therefore, we opt to employ additional objectives to represent uncertainty related risks.
4.2.1. Robust evolutionary optimization approaches

Two types of approaches for robust optimization was reported in Jin and Branke [22]: Single-objective and multi-objective. The first group integrates uncertainty handling as part of the fitness function. This could be done by randomly sampling and averaging the results as the fitness value. Another method is to inject noise into decision variables as optimization progresses. On the other hand, the second group captures uncertainty as additional objective(s). An additional objective function, for example, may represent the average value of original fitness function against a randomly sampled set of solutions adjacent to the solution represented by current individual.

One key benefit of Multi-objective robust optimization is that it provides rich decision-making support for product line manager. By presenting the results ranging from low-stability to high-stability, they can evaluate different solutions and perform risk analysis against the profit of each solution. This would support a spectrum of approaches varying from very pessimistic to very optimistic.

4.3. Approaches

In this section we introduce different approaches used in this study for modelling and implementing robust optimization. Table 2 shows a summary of all the approaches evaluated.

4.3.1. Evolutionary scoping optimization (ESO)

To establish a ground for explaining other approaches, we first explain a generic evolutionary algorithm shown in Algorithm 1. ESO employs a heuristic to find solutions that result in higher profit. In this case, uncertainty is not directly addressed.

In Algorithm 1 pseudo code, sp is the final result of optimization. There is one main loop that is repeated for a specific number of generations. selectPairs function returns a collection of pairs of two. This is later passed to crossover function to create new off-springs. mutate function performs mutation on off-springs. evaluate is where the performance of individuals are evaluated. This is done by passing the individual plans to objective function(s) and capturing the fitness value(s). Finally, selectBest function sorts and selects the best performing plans. These are set as the current population and will be used for the next generation.
Algorithm 1 ESO. The base approach in this study without explicit stability consideration.

\begin{verbatim}
sp ← randomly generated scoping plans
est ← estimates[0]
for generation in [1..n]
    pairs ← selectPairs(sp)
    offsprings ← crossover(pairs)
    newPlans ← offsprings + mutate(offsprings)
    evaluate(newPlans, est)
    sp ← selectBest(sp + newPlans)
next generation
\end{verbatim}

Algorithm 2 RSO uses a random sample of environment variables when evaluating plans.

\begin{verbatim}
sp ← randomly generated scoping plans
for generation in [1..n]
    pairs ← selectPairs(sp)
    offsprings ← crossover(pairs)
    newPlans ← offsprings + mutate(offsprings)
    est ← sample(estimate, 1)
    evaluate(newPlans, est)
    sp ← selectBest(sp + newPlans)
next generation
\end{verbatim}

4.3.2. Robust single-objective (RSO)

As stated before, one robust optimization approach found in literature is to run the optimization in a noisy environment. In this single-objective approach we let the environment change while the optimization is running. More robust solutions, compared to solutions with very volatile neighbourhood, are more likely to survive to the next generation. A pseudo code for this approach is given in Algorithm 2. In Algorithm 2, \texttt{sample} function picks one random sample of environment variables from the distribution of all possible environments.

4.3.3. Robust multi-objective with random sampling (RMO)

In their work, Marseguerra et al. [30] introduced the concept of Gradual Monte Carlo optimization. To evaluate the stability of an individual, a limited number of simulations are run and accumulated for each individual. This is repeated for each generation, when the individual is re-evaluated. The individuals that survive longer, would have a larger set of simulation cases and therefore have more accurate estimation of the actual stability.

Following a similar approach, to find the stable plans, we will be performing random sampling on the environment in each evaluation. In other words, there would be an additional step in the common evolutionary optimization algorithm.
Evolutionary optimization

Figure 4: Uncertainty sampling happens as an additional step in each iteration of an evolutionary algorithm.

Algorithm 3 RMO uses additional objective functions to measure the stability of plans.

\[
\begin{align*}
sp &\leftarrow \text{randomly generated scoping plans} \\
h &\leftarrow \text{sample size} \\
\text{for generation in } [1..n] & \\
&\quad \text{pairs }\leftarrow \text{selectPairs}(sp) \\
&\quad \ldots \\
&\quad \text{evaluate}(sp, \text{estimate}, h) \\
&\quad \ldots \\
\text{next generation} \\
\text{function evaluate } (sp, \text{estimates}, h) & \\
&\quad \text{for each plan in } sp \\
&\quad \quad \text{plan}.fa \leftarrow \text{profit(plan)} \\
&\quad \quad \text{plan}.fb \leftarrow \text{PVAR(plan, sample(\text{estimates},h))} \\
&\quad \quad \text{plan}.fc \leftarrow \text{PSWV(plan, sample(\text{estimates},h))} \\
&\quad \text{next plan}
\end{align*}
\]

(Figure 4). In our case, this would form two additional objective functions, \( f_b \) and \( f_c \) representing profit stability and feasibility stability respectively. For each individual, a sample set of environment \( Env \) of size \( h \) is created. The objective of the optimization is to maximize \( f_a \) while minimizing \( f_b \) and \( f_c \) subject to the constraints mentioned before. The pseudo code for RMO is given in Algorithm 3.

By solving this optimization problem, we will be trying to create solutions with maximum revenue while each individual may have varying level of robustness. This will give the product line decision maker the flexibility to trade-off optimality and robustness.

4.4. Stability measures

4.4.1. Profit stability (PVAR)

Since we are interested in changes in profit of the plan we are evaluating, one beneficial measure could be the variance of profit. In (8), \( Env \) is a collection of
samples of environment, $k$ is a single instance of this collection. Let’s assume

$$\text{ave}(x) = \frac{1}{|Env|} \sum_{k \in Env} f_a(x, k)$$

then the variance of profits would be:

$$f_b = \frac{1}{|Env| - 1} \sum_{k \in Env} (f_a(x, k) - \text{ave}(x))^2 \quad (8)$$

4.4.2. Feasibility Stability (PSWV)

To model the interest in solutions that are stable in terms of feasibility, we define $f_c$ as below (9). $v$ is a function that returns the number of constraint violations of $x$ in environment $k$.

$$f_c = \frac{1}{|Env|} \sum_{k \in Env} v(x, k) \quad (9)$$

4.5. Optimization Operators

For modelling scoping problem in a multi-objective heuristic search, we need to modify two components of a general-purpose search algorithm like NSGA-II. These components are a) encoding and b) objective function(s). In addition, due to having a custom encoding, generic operators do not work. Therefore, we will also need to implement our own crossover and mutation operators to achieve reasonable performance.

4.5.1. Encoding

In our encoding, each individual is represented using three arrays of integers. First array represents the product allocated to each segment. The values for each allocation should be either a product index or -1 which represents non-allocated segment. The second array represents the price of each product. Finally, third array represents the feature-build decisions. Possible values are: not-developed, reusable, and custom-developed. Encoding is demonstrated in Figure 5.

The benefit of this encoding compared to a conventional binary encoding is that it significantly reduces probability of generating invalid individuals by limiting the domain of values assigned to decision variables.

4.5.2. Crossover

Due to the custom encoding that we use for this study, we need to develop new crossover and mutation operators that are aware of such data structure. Our crossover operator starts by selecting two points $i \in [0..m)$ and $j \in [0..n)$. These will be used as partition points. The new off-springs resulting from the crossover operator would inherit $[0..i)$ segment allocation data from one parent and $[i..m)$ from the other. Similarly, the pricing is inherited from one parent for products $[0..j)$ and for products $[j..n)$ from the other parent. Feature-build is not explicitly involved. Instead, we use repair operator (explained below) to fix the feature-builds after crossover. Crossover operator is demonstrated in Figure 6.
Is the proposed approach effective for creating robust solutions?

As the main purpose of this study is to provide a method for creating scoping plans that are less sensitive to uncertainty, we would like to investigate the effectiveness of the proposed approach. This question can be divided into the questions below.

1. Is the proposed approach effective for creating robust solutions?
2. As a way to eliminate the issue with infeasible individuals, we opt for a repair method that fixes three types of constraint violations that could happen. Constraint violations in (6) are repaired by setting the segment allocation to the cheapest product found. Constraint violations in (7) are repaired by reducing the price. Missing feature constraint (4) is repaired by setting the decision of

4.5.3. Mutation

The mutation operator is also custom developed. Each mutation operation will either mutate the price of a single product, a single segment allocation or a feature-build decision. For price mutation, it adds or deducts a random percentage of the original price. For the segment allocation, it simply selects a segment and changes its assigned product to a random product. Similarly, feature-build decision mutation follows a similar approach. A random feature is selected and then its build decision is changed randomly. As the final step of both crossover and mutation, to avoid generating invalid individuals, we perform a repair on results of both operators.

4.5.4. Repair

As a way to eliminate the issue with infeasible individuals, we opt for a repair method that fixes three types of constraint violations that could happen. Constraint violations in (6) are repaired by setting the segment allocation to the cheapest product found. Constraint violations in (7) are repaired by reducing the price. Missing feature constraint (4) is repaired by setting the decision of
missing features to custom-developed. Finally, unnecessary features (5) are set to not-developed.

4.5.5. Selection
We use binary tournament as the selection operator [31]. This operator picks two solutions randomly from population and selects one that dominates the other solution in terms of objective function values. In case both solutions belong to non-dominated front, one of the two is randomly selected. The benefit of this operator is that it only uses the objective function results and therefore is reusable across problems with different encodings.

For this study, we use jMetal [32]. It is an optimization library that provides a number of common multi-objective algorithms including NSGA-II.

5. Research questions and study design

5.1. Research questions
Considering the research motivation and approach, we formulate our research into four research questions:

**RQ1:** How could uncertainty concerns be considered to guide SPL scoping optimization towards robust solutions? In other words, we would like a model that recognizes and also considers uncertainty as part of the optimization so that the decision variables are less susceptible to changes in estimated input variables. The answer to this question was previously given in Section 4. We presented RSO and RMO as two different approaches towards handling uncertainty. As a follow up to the previous question, we would like to know:

**RQ2:** What is the effectiveness of proposed approach for creating robust scoping plans? As the main purpose of this study is to provide a method for creating scoping plans that are less sensitive to uncertainty, we would like to investigate the effectiveness of approaches that target uncertainty and compare it with the cases where uncertainty is not considered. To evaluate our proposal, we need to investigate if our proposed approach performs better than other approaches? This would be the initial step to show the effectiveness of different existing approaches and also our proposal.

**Study design.** To evaluate this research question, we will compare three approaches: ESO, RSO and RMO. We will compare the robustness of results via simulation.

**RQ3:** How does the proposed approach perform under different uncertainty settings? To evaluate how different amounts of uncertainty affect the performance of optimization, we need to run evaluations under different settings for uncertainty.

**Study design.** To investigate this, we will perform all evaluations using three different level of uncertainty.
RQ4: How are the observed results affected by the multi-objective algorithm used? Evaluating the same problem using different algorithms and different settings would give insight about the performance and suitability of different algorithms for the current problem.

Study design. For this, we will compare different algorithms listed in Section 5.3.4. We will be using common measures like spread, hypervolume.

5.2. Data set
For this study, a dataset needs to contain values for the following input estimates:

- Willingness-to-pay: What are the customer segments and how much are they willing to pay for each product configuration?

- Candidate products: What are the candidate product configurations? For example, a product line may have three candidates with specific features.

- Feature costs: What is the development cost of a feature? Specifically, our model expects estimates about three different costs: 1. the cost of uniquely developing a feature for a specific product, 2. The cost of developing a feature with reusability in mind and 3. the cost of reusing a feature in a product.

- Competitor pricing: What is the estimated price of an offering from a competitor for a specific segment?

Additionally, anticipated amount of inaccuracy in estimates over willingness-to-pay, feature cost, and competitor pricing is required for sampling process. To summarize, the expected dataset needs to be represented in a similar data model as depicted in Figure 7.

5.2.1. Home automation system
The home automation system dataset is adopted from Alswalqah et al. [33]. This dataset has 36 features, 5 segments and one product for each segment. For product-feature assignment, we used the optimization results of that study (optimized for cost saving). As it does not contain willingness-to-pay, competitor pricing and uncertainty data, we added willingness-to-pay and competitor pricing randomly by performing some experiments and finding a combination that was at least challenging for the optimizer. For this dataset, we also assumed each segment has one customer. For uncertainty, see Section 5.4.1.
5.3. Experiment design

5.3.1. Evolutionary algorithms parameters

To make the comparison between different approaches possible, we need to pick a number of parameters shared between evolutionary algorithms. The following are selected based on their performance in our empirical evaluations and because they are commonly used in the literature. The crossover and mutation probability for this study is set to 0.9 and 0.05 respectively. Total number of evaluations for each run is 10000. Population size was set to 100. We run each setting a number times to reduce the effect of randomness. Number of runs was 5 or 10 depending on the type of experiment and its sensitivity to randomness. To create the results-sets used for comparison and analysis, we need to aggregate different runs into a single result set. To do this, we use the non-dominated front of all runs.

5.3.2. Comparing ESO, RSO and RMO

This experiment is designed to answer research questions RQ2 and RQ3. Often when algorithms generate Pareto fronts with equal number of dimensions, measures like hypervolume or spread are used to compare the approaches. In case of this experiment, the single-objective approach does not generate a front but only one solution. This renders measures mentioned above inapplicable. Another challenge has to do with size of the problem and the areas that algorithms converge to. Since the single-objective operates in a solution space of lower dimensionality, it finds solutions with higher profit while the multi-objective one is more likely to avoid those solutions due to having poor stability. In summary, the attained results may have differing profit and therefore comparing stability of those solutions may not be fair.

Target profit analysis. To mitigate the issue mentioned above, and to be able to better compare the different approaches, we need to somehow direct the search towards solutions with similar profit. This way, we would be able to evaluate and compare the stability of results. For this, we propose to change the profit objective function \( f_a \) so that instead of maximizing, the algorithm tries to minimize the distance to a specific value. The result is the \( f'_a \) defined as:

\[
f'_a = |f_a - tp|
\]  

In (10), \( tp \) is a target profit that the optimization will try to converge to. For each \( tp \) we report the average of all stability measures and also Mann-Whitney U-test results to investigate if the observed difference is meaningful or not. We also use Vargha-Delaney’s A measure to show the significance of the difference.

5.3.3. Random sampling strategies

One drawback of random sampling is that if the initial set of samples for a stable individual is not promising, it may loose the chance to survive the first round of evaluation. Therefore, thoughtful selection of sampling settings is important [30]. To evaluate this we experiment with different sample sizes and sampling strategies.
5.3.4. Comparing different algorithms

This experiment is designed to answer question RQ4. The following lists the algorithms used in our study:

**NSGA-II [13]** In this algorithm solutions are sorted based on their dominance. Non-dominated solutions are selected for next generation. Among the solutions within the same front, solutions resulting to a better uniformly distributed front are preferred. This is achieved using crowding distance.

**SPEA2 [14]** This algorithm takes a different approach at calculating the fitness of a solution. Fitness is a combination of fitness function values and also the distance of that solution to a specific neighbourhood. Non-dominated solutions are stored in an external set called archive.

**MOCell [16]** This algorithm uses the concept of near neighbourhood. When performing crossover, only the near neighbourhood of a solutions is considered rather than the whole population. This is to achieve better exploration by avoiding early convergence. To store the non-dominated front, similar to SPEA2, solutions are stored into an archive which also uses crowding distance to help diversity of the front.

NSGA-II is selected for this study because it is among the most referenced multi-objective algorithms in the literature. SPEA2, on the other hand uses a different method to manage diversity compared to NSGA-II based on nearest neighbourhood. Finally, MOCell is among the most recent algorithms proposed in the literature. Konak et al. [34] reported a comprehensive discussion and comparison of different multi-objective optimization algorithms.

5.4. Empirical evaluation and statistical analysis

The results of non-deterministic algorithms may differ between different runs. For example for multi-modal problems, per each run, the algorithm may converge to a different optimum. Additionally, the algorithm may not be able to find the optimum but can get very close to such point. This behaviour makes evaluation challenging. In our work, we are using search-based algorithms that also fall under non-deterministic algorithms and therefore the above behaviour applies to current study as well.

Arcuri and Briand [35] proposed a guideline for how to design studies that employ non-deterministic algorithms for software engineering problems. Based on their recommendation, we follow the following for the experiments in this study:

- Clearly report the algorithm parameters. For example number of runs, population size, etc.
- Run each algorithm with a specific setting for multiple times to mitigate the effect of randomness.
• For statistical analysis we use Mann-Whitney U-test to compare the results of two approaches. This will check if the results are significantly different or the difference is due to displacement. Mann-Whitney U-test is recommended in the literature because of being a non-parametric test that, unlike t-test, does not require the sample to belong to a population with a specific distribution.

• For comparison, we also want to know the extend of difference in results. One simple way to do this is to compare the average of two distributions. This would not be very accurate if the distribution has high variance. One better way to do this is via effect size analysis. In this study, we use the Vargha-Delaney A measure.

• We try to demonstrate the results using box-plots and other descriptive methods.

Statistical tests were performed using R and pertaining libraries.

5.4.1. Simulation for sensitivity analysis

To study the impact uncertainty, we use an approach similar to the one introduced by Cantor [10]. For each individual, we create 200 variations of the original environment and re-evaluate all objective functions in these new environments. For this, we create new environments by applying the output of a random function to the original environment.

Estimation uncertainty distribution is either extracted empirically from previous data or by consulting experienced software engineers. Because of its simplicity and efficiency, triangular distribution is used for representing uncertainty [36, 37]. In our study, we use triangular probability distribution. We assume δ is the percentage of deviation from original estimate and for triangular distribution lower limit, mode and upper limit are set to −δ, 0 and δ respectively. Note that study design is not dependent on a specific distribution and triangular distribution could easily be replaced by, for example, normal distribution without affecting the study design. To model different level of uncertainty in estimates, we define three values for representing low, medium, and high uncertainty respectively (δ ∈ {2.5, 15, 30}).

To be able to investigate the quality of results, we use the PVAR and PSWV measure introduced in Section 4.4.1 and Section 4.4.2.

6. Results and discussion

In this section, we report the results of different experiments. A summary of experiments, their settings and parameters are reported in Table 3.

6.1. Comparison of ESO, RSO and RMO

In this section we report the results of two sets of experiments. First we employ the alternative profit objective function to target different profits. Next we employ the original objective function that maximizes the profit.
Experiment Approach Algorithm
Target profit analysis (6.1) ESO, RSO, RMO NSGA-II
Maximum profit (6.1.2) ESO, RSO, RMO NSGA-II
Sampling sizes (6.2) RMO NSGA-II
Sampling methods (6.3) RMO NSGA-II
Base algorithm (6.4) RMO NSGA-II, SPEA2, MOCell

6.1.1. Target profit
In this experiment $f'_a$ is used instead of $f_a$ to guide the optimization to specific profit values. This is done for profits of \{5000, 10000, 15000, 20000, 25000, 35000, 40000\}. Simulation results are reported in Figure 8. As depicted, for $\delta = 2.5$ (Low), RMO was able to find solutions that have relatively lower PVAR and PSWV. The reason for this is that the search space for such condition is very relaxed and the probability of finding a stable solution in this search space is higher. Although, as we move to higher target profits, finding solutions becomes more challenging. For ESO, an increase in both measures can be observed. On the other hand, RMO tend to achieve better PSWV even with very high target profits. Finally, the results for RSO show that as we move towards higher target profits, the number of feasible solutions in the population decreased. This decrease is because the high changes in environment variables and the fact that RSO evaluates a solution using only one random sample.

For experiments with $\delta = 15$ (Med.), optimization becomes more challenging. This is observable by the increase in the number of not-close solutions generated
Table 4: Target profit test results

<table>
<thead>
<tr>
<th>tp</th>
<th>Appr.</th>
<th>Measure</th>
<th>P-Value</th>
<th>Effect size</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>ESO</td>
<td>PSWV</td>
<td>0.275</td>
<td>0.471</td>
</tr>
<tr>
<td>5</td>
<td>ESO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.985</td>
</tr>
<tr>
<td>5</td>
<td>RSO</td>
<td>PSWV</td>
<td>0.257</td>
<td>0.549</td>
</tr>
<tr>
<td>5</td>
<td>RSO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.993</td>
</tr>
<tr>
<td>10</td>
<td>ESO</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.316</td>
</tr>
<tr>
<td>10</td>
<td>ESO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.938</td>
</tr>
<tr>
<td>10</td>
<td>RSO</td>
<td>PSWV</td>
<td>0.040</td>
<td>0.584</td>
</tr>
<tr>
<td>10</td>
<td>RSO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.785</td>
</tr>
<tr>
<td>15</td>
<td>ESO</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.990</td>
</tr>
<tr>
<td>15</td>
<td>ESO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.998</td>
</tr>
<tr>
<td>15</td>
<td>RSO</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.991</td>
</tr>
<tr>
<td>15</td>
<td>RSO</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.994</td>
</tr>
<tr>
<td>20</td>
<td>ESO</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.862</td>
</tr>
<tr>
<td>20</td>
<td>ESO</td>
<td>PVAR</td>
<td>0.033</td>
<td>0.679</td>
</tr>
<tr>
<td>20</td>
<td>RSO</td>
<td>PSWV</td>
<td>0.002</td>
<td>0.776</td>
</tr>
<tr>
<td>20</td>
<td>RSO</td>
<td>PVAR</td>
<td>0.003</td>
<td>0.767</td>
</tr>
</tbody>
</table>

by RMO. These solutions, depicted using black circles, are solutions that are not within the $[tp - \frac{5}{\delta}, tp + \frac{5}{\delta}]$ range. RMO favours stable solutions by trading-off profit. On the other hand, other two approaches tend to find solutions with worse stability. This is specially observed for solutions with the target profit of 20000 and 25000.

Finally, for $\delta = 30$ (High), both ESO and RSO start to struggle finding stable solutions. Most solutions found by these approaches have very high PSWV and PVAR. RMO performs better with lower target profits. Again, the increase in number of black circles depicts the trade-off happening between profit and stability. For very high target profits, it also starts to struggle but still manages to find solutions with better stability (both close and not-close to target profit). Looking at $tp = 10000$, its interesting to see that although RMO has heuristics to guide the optimization towards better stability, other two approaches generate solutions with better PSWV. This could be explained by the trade-off happening between the two stability objective functions (rather than profit). That is, it seems like for this specific target profit, finding solutions with lower PVAR is more likely than solutions with better PSWV.

To better depict how different approaches compare to each other, averages of two measures along with their standard error are depicted in Figure 9 and 10.

To evaluate the results reported in Figures 9 and 10, the statistical test results of this experiment are presented in Table 4. Note that due to the large number of test cases, only the results for an uncertainty of $\delta = 15$ are reported. Also note that for each test, ESO or RSO is compared to RMO.

6.1.2. Maximum profit

In this experiment, original $f_a$ is used. This lets the optimization to find solutions with highest possible profit. Figure 11a shows the distribution of profits across three approaches and three different $\delta$. As demonstrated, for $\delta \in 2.5, 15,$
Figure 9: Average values for all solutions

ESO and RSO tend to find higher profits. But in a very volatile environment ($\delta = 30$), RSO starts to avoid those solutions. Interestingly, for all different levels of uncertainly, RMO tends find a wider spread of profits. The reason for this behaviour is depicted in Figure 11b showing the average PVAR and PSWV measures. It is also depicted that RMO, on average, finds solutions with lower PVAR and PSWV.

6.2. Different sampling sizes for RMO

In this experiment we will be evaluating the effect of sampling size on results of RMO. Table 5 shows the statistical test results of three different samples sizes:
As depicted in Table 5, there is a significant difference between the result with sample size of 1 and 15. This is expected as the size of random sampling translates to the confidence we have in PSWV and PVAR estimates. When comparing the results of size 15 to 30, the difference is not observed for PVAR measure.

To see how good sample sizes translate to good estimates, we define two simple measures: Profit variation risk error and Feasibility risk error. Both are defined as the difference between simulation result and the stability objective function value. Figure 12a and Figure 12b show the distribution of these error
(a) Kernel density estimate when targeting maximum profit

(b) Average PSWV and PVAR when targeting maximum profit

Figure 11: Maximum profit analysis

<table>
<thead>
<tr>
<th>Uncertainty</th>
<th>Sample</th>
<th>Measure</th>
<th>P-Value</th>
<th>Effect size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>1, 15</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.993</td>
</tr>
<tr>
<td>Low</td>
<td>1, 15</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.735</td>
</tr>
<tr>
<td>Low</td>
<td>1, 30</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.993</td>
</tr>
<tr>
<td>Low</td>
<td>1, 30</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.703</td>
</tr>
<tr>
<td>Low</td>
<td>15, 30</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.41</td>
</tr>
<tr>
<td>Low</td>
<td>15, 30</td>
<td>PVAR</td>
<td>0.031</td>
<td>0.441</td>
</tr>
<tr>
<td>Med.</td>
<td>1, 15</td>
<td>PSWV</td>
<td>0.000</td>
<td>1</td>
</tr>
<tr>
<td>Med.</td>
<td>1, 15</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.715</td>
</tr>
<tr>
<td>Med.</td>
<td>1, 30</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.999</td>
</tr>
<tr>
<td>Med.</td>
<td>1, 30</td>
<td>PVAR</td>
<td>0.000</td>
<td>0.643</td>
</tr>
<tr>
<td>Med.</td>
<td>15, 30</td>
<td>PSWV</td>
<td>0.000</td>
<td>0.392</td>
</tr>
<tr>
<td>Med.</td>
<td>15, 30</td>
<td>PVAR</td>
<td>0.122</td>
<td>0.454</td>
</tr>
<tr>
<td>High</td>
<td>1, 15</td>
<td>PSWV</td>
<td>0.015</td>
<td>1</td>
</tr>
<tr>
<td>High</td>
<td>1, 15</td>
<td>PVAR</td>
<td>0.699</td>
<td>0.419</td>
</tr>
<tr>
<td>High</td>
<td>1, 30</td>
<td>PSWV</td>
<td>0.015</td>
<td>1</td>
</tr>
<tr>
<td>High</td>
<td>1, 30</td>
<td>PVAR</td>
<td>0.652</td>
<td>0.406</td>
</tr>
<tr>
<td>High</td>
<td>15, 30</td>
<td>PSWV</td>
<td>0.13</td>
<td>0.545</td>
</tr>
<tr>
<td>High</td>
<td>15, 30</td>
<td>PVAR</td>
<td>0.368</td>
<td>0.927</td>
</tr>
</tbody>
</table>

Table 5: Statistical test results of three sampling sizes

values. As depicted, for both sample sizes of 15 and 30, the PSWV distribution is skewed to the right while for 1, it is skewed to the left. This means that sample size of 15 and 30 yield more accurate results with smaller error. For PVAR the difference is not as significant but still the distribution of sample sizes of 15 and 30 are heavily skewed to the right. It is interesting to see that a sample size of 15 in both cases manages to find most scoping plans with error of zero. This shows that with sample sizes as small as 15, satisfactory results are attainable.
6.3. Different sampling methods for RMO

In this experiment, we evaluated different sampling methods for RMO. These sampling methods are:

**Once:** Environment is sampled only once, when the solution is evaluated.

**Every selection:** Environment is sampled every time a solution is selected using selection operator.

**Maturing:** Uncertainty is sampled during generations in a descending manner. That is, sampling is done more often in the initial generations compared to later ones.

**Increasing:** Uncertainty is sampled only a few times initially but for more number of times as generations passes. This is the opposite of maturing approach.

**Selective:** Uncertainty is sampled only if the previous sampling show that the solution does not have either very high or very low uncertainty. This is motivated by the fact that samples with congested uncertainty (either high or low) have higher probability of representing the population.

As depicted in Figure 13, performing sampling in each selection results in better stability while performing fewer sampling in the beginning and increasing the number of samplings with later selections results is less stable solutions but with higher profit.

6.4. Effect of base algorithm on robust optimization

In this experiment we evaluated the effect of using a different base algorithm for the RMO approach. For this we used the algorithms listed in Section 5.3.4.

The following quality indicators were used to evaluate the results.

**Spread** This quality indicator is used for measuring the distance between solutions. Values closer to zero represent a better spread front while higher values show that the solutions may not be as evenly distributed [13].
Figure 13: Comparison of different sampling methods. Both measures are scaled to facilitate comparison.

**Hypervolume** Spread does not compare the actual values of objective functions directly. Hypervolume is an indicator that models the non-dominated front as a volume. To calculate this measure, a reference point is needed to represent the worst attainable solution. Each solution from the solution set and the reference point would form a volume (a hypercube). The volume of the union of all these hypercubes is called hypervolume. Higher values for this measure represent solutions sets with higher quality and therefore are preferred [16].

**Epsilon** $\varepsilon$ represents the smallest distance between two fronts. Similar to Hypervolume, $\varepsilon$ indicator requires a reference point which is generally the Pareto optimal front. If this front is not known, any reference point could be used [38]. We use the best front of all runs for this purpose.

Results of this experiment is reported in Figure 14. As depicted by the amount of overlap for hypervolume and $\varepsilon$, there is generally no significant difference between the three evaluated algorithms while MOCell achieves better spread. This is inline with reports from the literature and supports the argument that the effect observed in other experiments is not a result of choice of base algorithm.

6.5. *Answers to research questions*

**RQ1 and RQ2.** To answer the research questions RQ1 and RQ2, we started based on an existing model and added extensions to capture uncertainty as soft-constraints using additional heuristics (RMO). Additionally, we implemented an
approach that would allow noise to be present while the optimization is running (RSO). Finally, to represent the baseline for the analysis, an approach with no explicit support for uncertainty was also implemented.

In RMO, uncertainty was captured using two additional heuristics, one representing the stability of profit and the other representing the feasibility stability. Our evaluations show that although additional objectives increase the dimensions and hence the complexity of the problem, RMO was able to find reasonable number of solutions spread across the Pareto front. Specially when profit is considered as the main objective, RMO managed to find solutions with as high profit as the other two approaches while also maintaining higher stability.

**RQ3.** For the effect of uncertainty, we considered three levels of uncertainty as described in Section 5.4.1. Results reported in Sections 6.1 show that RMO results in solutions with better stability even with very high uncertainty levels. As depicted in the target-profit experiments, when target-profit is increasing RMO still results in significantly more stable solutions.

**RQ4.** We designed and implemented our approach without relying on specific behaviour of a multi-objective algorithm. Although the experiments to answers all previous research questions were performed using NSGA-II (Table 3), we also evaluated the effect of base algorithms on performance. As discussed in Section 6.4, SPEA2 and NSGA-II, achieved similar performance across three levels of uncertainty with no significant difference while MOCell achieved better spread. This indicates that the observed results in previous research questions are not...
the result of base algorithm (NSGA-II) and we expect to attain similar results when this approach is implemented using other similar algorithms.

6.6. Threats to validity

Internal validity. We have made a number of decisions to increase internal validity. First, since all of the evaluated approaches were non-deterministic, we repeated each run for a number of times to reduce the effect of randomness. As for optimization parameters, like crossover and mutation probability, we picked values that are common in literature. We also did not perform extensive study on these parameters and do not argue that our choice is preferred as this is outside the scope and purpose of this study.

To design studies, we tried to follow recommended guidelines from the literature specially about evaluating randomized algorithms. We reported statistical tests and to address the concern of alternative cause, we performed extensive experiments and evaluations.

External validity. In this study we had to make assumptions about the model and the dataset. For random sampling, simulation and to model uncertainty, we picked a distribution model that is simple for modelling and also had been used in literature before. Uncertainty in real-world projects may have a different distribution than what we assumed. We also made assumptions about the modelling as well. We assumed that product configuration is available before hand. That is, we know what are the specific candidate products that we may want to consider and what their features are. An alternative model for scoping may include feature selection and product configuration as part of the model to relax this assumption. Finally, to be able to generalize this study to real-world projects, we chose a relatively large span for uncertainty ranging from very low (±2.5%) to high (±30%).

Complexity wise, one drawback of RMO is the cost of additional computation. When evaluating individuals, to calculate the values of the two stability objective functions, we need to sample the environment variables. The extra cost associated with this sampling would increase when larger sample sizes are selected. To mitigate this additional cost, selection of sample size requires careful consideration.

6.7. Related works

From the general software engineering perspective, Mkaouer et al. [39] model robust refactoring using multiple objectives and perform trade-off optimization. They evaluated their proposal using different algorithms including NSGA-II. Similarly Gueorguiev et al. [40] modelled project planning as a multi-objective problem with one main objective representing the total time required to complete tasks and two additional objectives representing robustness, both also based on time. Their target was to reduce time while still creating robust plans that consider unexpected delays. They used SPEA2 algorithm and evaluated it against a simple random algorithm. To represent uncertainty, they randomly
added new planning items and also increased the estimated cost of existing tasks. Although, similar to our approach, both works use additional objectives to support robustness, they only consider performance stability while our approach considers feasibility stability as well.

*Search-based software engineering.* Recently, there has been an increase in applying search-based methods to software engineering problems. As for SPLs, Lopez-Herrejon et al. [41] reports a literature review on search-based methods. The findings indicate that most studies concentrate on issues like requirements engineering, application-engineering, maintenance and evolution. In our study, we addressed SPL scoping optimization which has not received as much attention as other issues.

*Dynamic optimization.* Dynamic problems share many similarities to robust optimization. In dynamic problems, the optimization is expected to respond to changes to assumptions while optimization is running and is not completed. Bui et al. [42] reported on application of multi-objective approaches for solving dynamic single-objective problems. Their approach uses an additional objective to create a more diverse solution-set. They found that approaches that search for solutions based on their distance to other members of solutions-set perform better. In comparison, our approach generates solutions that stay stable after optimization is completed. In other words, scoping plans can tolerate changes in environment without significant change in profit and loss of feasibility.

*SPL scoping optimization.* Müller [7] provided a cost/value analysis method toward product line scoping. It used heuristics to find the optimal settings for the product line. The results of optimization are price recommendation and segment allocation. Alsawalqah et al. [33] modelled scoping as a cost minimizing problem that try to increase the number of features developed for reuse. In both studies, uncertainty was not considered.

### 7. Conclusions and future Work

This paper presented a robust optimization approach for software product line planning activities including portfolio scoping and asset scoping. In this work, we incorporated uncertainty into SPL scope modelling and performed optimization by simulating changes in environment. The factors representing robustness are profit stability and feasibility stability.

Our model was implemented using a multi-objective evolutionary approach. We showed that there are alternative scoping plans that, compared to others, can tolerate estimation inaccuracies and changes in environment. Our approach searched for these plans by guiding the optimization towards solutions that have higher stability and avoiding those that are prone to uncertainty. Also, results showed that there is a trade-off between profit and robustness. Increase in profit would decrease the number of feasible scoping plans that are relatively stable.
We foresee a number of future works as follow-ups of this study. As the final goal is to enable a product line manager with enhanced decision support, one route for future research is to investigate the effectiveness of this approach from their perspective.

Although product portfolio scoping is supported by the model presented in this paper, one important aspect of the SPL scoping that is not covered is the support for generating product portfolio candidates or feature selection. This area of research has recently received a lot of attention. Integration of feature selection and current study may be an interesting topic.

With regard to pricing strategies, in our approach, we assume that a product is marketed with a single price to customers. In contrast, there are approaches that provide personalized pricing. For example, Schön [43] described a profit maximization method that offers price customization. Consideration of other pricing strategies may also be a possible future work.

As for optimization itself and to achieve better performance, there have been efforts towards joining evolutionary algorithms and local search [44] from which this study can benefit.

8. Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant 250343-12 and Alberta Innovates Technology Futures award.

References


