Elise Vareilles Chiara Grosso
José Miguel Horcas Alexander Felfernig (Eds.)

ConfWS 2024
26th International Workshop on Configuration

Girona, Spain, September 2-3, 2024
Proceedings

Editors' addresses:
Universitat de Girona
Edifici Les Àligues
17071 Girona, Spain

elise.vareilles@mines-albi.fr, chiara.grosso@uniroma1.it, horcas@uma.es, alexander.felfernig@ist.tugraz.at

Preface

The 26th edition of the International Workshop on Configuration (ConfWS 2024) has been co-located with the International Conference on Principles and Practice of Constraint Programming (CP 2024 celebrating its 30th anniversary) hosted by the University of Girona in Spain. ConfWS 2024 has been a vibrant hub for researchers and industry professionals interested in configuration technology. In addition, for edition 2024, Siemens supported the event confirming its sponsorship.

ConfWS 2024 was a two-day event where high-quality research in all configuration-related technical areas has been presented. This edition had a special focus on Green Configuration which is related to EU Green Deal as stated in the EU Agenda 2050 to drive the EU community to a more sustainable future. Researchers and experts from academia and industry shared their contributions on the potentials of configuration in achieving sustainability goals for a more sustainable future. The program includes special sessions on green configuration and sustainability, including topics such as sustainability and configurator applications, efficient reasoning, configuration space learning, integration of large language models (LLMs), and further aspects related to problem solving and optimization.

ConfWS 2024 has been visited by 24 attendants from academia and industry. There were 14 papers submitted for peer review to ConfWS 2024. 14 papers were selected for publication in the workshop proceedings after a review by three independent reviewers per paper. In addition, three keynote speakers were invited from three industrial partners: Patrik Östberg and Sonja Arce (from Tacton) presented the talk "Inspiring & enabling manufacturers to shape and build a sustainable future"; Sophie Rogenhofer (from Siemens) presented the talk "Sustainability at Siemens - Scaling sustainability impact"; and Jean-Guillaume Fages (from Cosling) presented the talk "Automating complex computations with Cosling Configurator".

ConfWS 2024 introduced the role of a "Publicity and Social Media Chair" on the organization committee, a role played by Irene Campo Gay (Technical University of Denmark), who was in charge of promoting the workshop on social media and managing the official workshop accounts on LinkedIn and Twitter (X). In line with previous editions, the workshop participants selected the best paper ("Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems") and the best student paper ("Configuration Copilot: Towards Integrating Large Language Models and Constraints").

We want to thank the ConfWS 2024 authors for their high-quality submissions, the program committee members for their high-quality reviews, and the University of Girona and the CP Workshop Chair and CP Chairs for their proactive support. Further thanks goes to SIEMENS for sponsoring ConfWS 2024, and the keynote speakers for delivering inspiring presentations. The following projects by the Spanish Ministry of Science, Innovation and Universities also supported the workshop: *TASOVA PLUS* research network (RED2022-134337-T), *IRIS* (PID2021-122812OB-I00), and *Data-pl* (PID2022-138486OB-I00).

September 2024

Elise Vareilles, Chiara Grosso
José Miguel Horcas, Alexander Felfernig

Workshop Chairs

Elise Vareilles, IMT Mines Albi, France
Chiara Grosso, DIAG, University La Sapienza Rome, Italy
José Miguel Horcas, Universidad de Málaga, Spain
Alexander Felfernig, Graz University of Technology, Austria

Publicity and Social Media Chair

Irene Campo Gay, Technical University of Denmark, Denmark

Award Chair

Alexander Felfernig, Graz University of Technology, Austria

Program Committee

Gerhard Friedrich, Alpen-Adria-Universität Klagenfurt, Austria
Gerhard Leitner, Alpen-Adria-Universität Klagenfurt, Austria
Lothar Hotz, Hamburger Informatik Technologie-Center, Germany
Andreas Falkner, Siemens AG Österreich, Vienna, Austria
Lidia Fuentes, Universidad de Málaga, Spain
Yue Wang, Hang Seng University, Hong Kong
José Ángel Galindo, Universidad de Sevilla, Spain
Enrico Sandrin, University of Padova, Italy
Markus Stumptner, University of South Australia, Australia
Lars Hvam, Technical University of Denmark, Denmark
David Benavides, Universidad de Sevilla, Spain
Cipriano Forza, University of Padova, Italy
Albert Haag, Product Management GmbH, Germany
Alois Haselboeck, Siemens AG Österreich, Austria
Richard Comploi-Taupe, Siemens AG Österreich, Vienna, Austria
Tomi Männistö, University of Helsinki, Finland
Viet-Man Le, Graz University of Technology, Austria
Abdourahim Sylla, Université Grenoble Alpes, France
Juha Tiihonen, Variantum, Finland
Franz Wotawa, Graz University of Technology, Austria
Rüdiger Dehn, Lino GmbH, Germany

Contents

Configuration of Heterogeneous Agent Fleet: a Preliminary Generic Model

Thomas Pouré[1,†], Stéphanie Roussel[2,†], Elise Vareilles[1,3,*,†] and Gauthier Picard[2,†]

[1] ISAE SUPAERO, Université de Toulouse, 10 avenue Édouard Belin, BP 54032 - 31055 Toulouse CEDEX 4, France
[2] DTIS, ONERA, Université de Toulouse, 2 avenue Édouard Belin, BP 74025 - 31055 Toulouse CEDEX 4, France
[3] CGI / IMT Mines Albi, Université de Toulouse, allée des sciences, 81000 Albi, France

Abstract

A multitude of autonomous agents – encompassing a range of technologies, including robots and drones – represent a crucial modern tool for the execution of a multitude of tasks, including surveillance, delivery and the saving of lives. In order to optimally utilise these agents, it is vital to configure each agent, the composition of the entire fleet of agents and the mission plan associated with each agent in the most effective manner possible. The following article presents a knowledge model for the configuration of a fleet of heterogeneous agents, encompassing the three levels of configuration: agent configuration, agent fleet configuration, and mission plan configuration. It explicitly delineates the relationships between these three configuration levels, thereby facilitating rapid, efficient, robust, and simultaneous configuration. A toy problem illustrates our first proposals.

Keywords

Multi-level Configuration, Autonomous Agent, Knowledge Formalisation, Heterogeneous Fleet,

1. Introduction

With the increasing autonomy of drones and robots, fleets of agents are now being used for many different types of missions, such as exploration, rescue, disaster relief, civil and military security. In this article, the term "agent" is used to refer to any system that is capable of acting autonomously in a variety of environments, including ground, water, and air. The term encompasses a diverse range of platforms, including quadrupeds, bi-blades, underwater rockets, and others. Additionally, the term "agent" encompasses a wide range of capabilities, including communication, rescue, and delivery. Therefore, the term "agent" can be used to describe a diverse range of systems, from household robots to high-tech stealth military drones. Some of these applications require heterogeneous agent fleets, i.e. with different platforms, capabilities, mobility and equipment. Such fleet of heterogeneous agents may or may not be coordinated autonomously to carry out the missions to which the fleet is dedicated. For example, an exploration mission may require the collaboration of ground agents with at least the ability to *Travel* and *Communicate*, and aerial agents with at least the ability to *Observe* and *Communicate*. The success of a multi-agent mission depends, among other things, on the configuration of the fleet executing it [1].

This paper addresses the problem of *multi-level configuration of heterogeneous agent fleets*, as presented in Fig. 1. By multi-level configuration, we mean the several interleaved problems that must be solved when setting up a fleet to carry out a mission. The first level is the simultaneous configuration of each agent (Agent Configuration Problem, ACP). The second consists in configuring the fleet itself (Fleet Configuration Problem, FCP), i.e. defining precisely what the composition of the fleet is. The final level is the fleet deployment problem in order to carry out dedicated missions in an efficient and robust way (Plan Configuration Problem, PCP). This multi-level configuration problem requires an analysis of the relationships between these three configuration levels, both upstream in fleet composition and downstream in fleet operation.

This multi-level configuration problem raises many research questions, such as:

- the representation/modeling of configuration knowledge (compact modeling language),
- eliciting constraints (what is allowed or forbidden) and criteria (what is preferable) that apply both to the fleet configuration and to each robot in it, and
- the development of algorithms to generate optimal or, at least, good-quality solutions.

This problem can be tackled in several ways. First of all, there is the question of how to express knowledge,

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
†These authors contributed equally.
thomas.poure@student.isae-supaero.fr (T. Pouré); stephanie.roussel@onera.fr (S. Roussel); elise.vareilles@isae-supaero.fr (E. Vareilles); gauthier.picard@onera.fr (G. Picard)
https://onera.academia.edu/SRoussel (S. Roussel); https://pagespro.isae-supaero.fr/elise-vareilles/ (E. Vareilles); https://gauthier-picard.info/ (G. Picard)
0000-0001-7033-555X (S. Roussel); 0000-0001-6269-8609 (E. Vareilles); 0000-0002-9888-9906 (G. Picard)

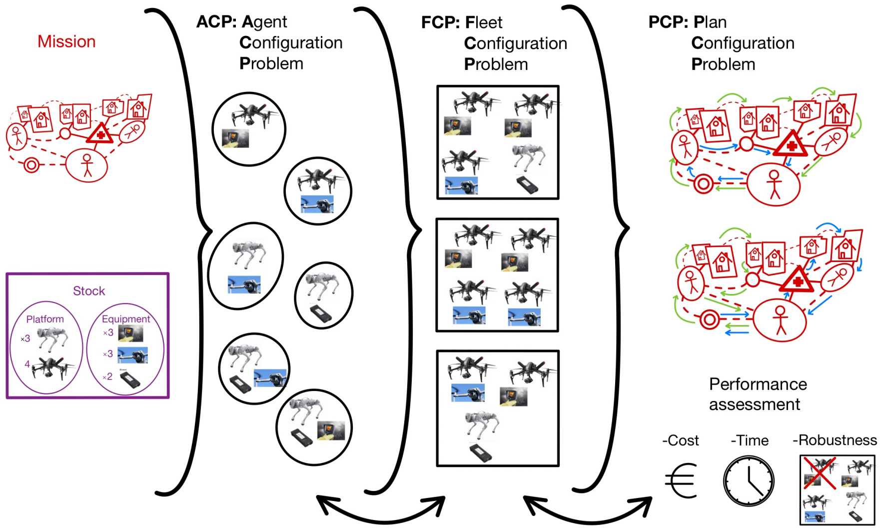


Figure 1: Multi-level configuration of an heterogeneous fleet of robots.

constraints and preferences, both from the point of view of fleet configuration and from the point of view of performance and robustness in the context of mission [2]. Approaches such as constraint programming and multi-agent modeling [3, Chap.2 and 15] appear to be suitable candidates.

Following several works dedicated to Search and Rescue applications such as [4] and [5], a mission consists here in the execution of several tasks distributed on an intervention zone represented by a graph. A fleet and a plan of action are configured in order to accomplish the mission, i.e. successfully complete all the tasks. The performance of a fleet for a mission can be evaluated along several criteria: the global time required for performing all tasks, the fleet cost (platform, equipment), the fleet and the plan robustness (capacity of the fleet/the plan to support damages and complications), etc.

This article focuses on initial ideas for modeling the knowledge of this multi-level configuration problem of heterogeneous agents fleets. More precisely, we propose a formal modelling of the inputs of each level configuration problem, along with the decisions that have to be made. The formalization of constraints associated with each level are out of scope of this paper and are left for future work.

The paper is organized as follows. In Section 2, we formally describe the type of mission we consider. Then, Sections 3, 4 and 5 are respectively dedicated to the Agent Configuration Problem or ACP, the Fleet Configuration Problem or FCP and the Plan Configuration Problem or PCP. In each of these sections, we formally present the inputs of the problem, the associated decision variables and an illustrative example. Finally, we conclude and discuss future works in Section 6.

2. Mission

A mission allows to represent the several tasks that the agents have to perform and the graph on which they can move. The elements composing a mission can be represented in a UML diagram as illustrated in Fig. 2. Those elements are first briefly described and then formalized in a second step. In our work, we have made several assumptions on a mission. A mission is therefore:

- **deterministic**: the mission is perfectly known from the beginning and during the fleet's intervention, and agents cannot suffer from malfunctions,
- **static**: the mission remains static throughout the fleet's intervention. No edges or vertices are introduced or removed during the mission.

2.1. Description

A Mission is composed of the following elements.

- The location on which the agents can evolve is represented by a connected and non-directed Graph. Such a graph is composed of vertices (Vertex class), representing way points or places of interest in the mission context, and edges (Edge class), representing routes for moving.

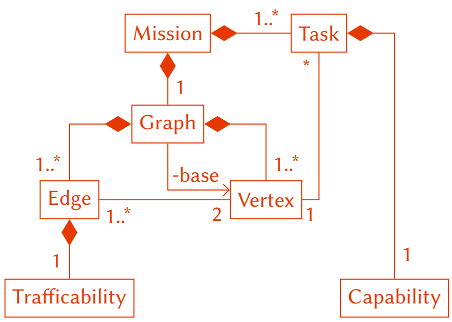


Figure 2: UML representation of a mission.

- One of the vertices is called the base, and is the location at which the agents start and finish their missions.
- The actions that the agents have to perform to achieve the mission are called Tasks. Each task is assigned to a single vertex, that represents the location at which it must be executed. A capability is associated to each task, it is the requirement to perform a task.

For the agent to be able to move through the graph and execute task, we define two additional classes.

- A Capability describes how an agent accomplishes the mission's tasks. More precisely, each task requires a single specific capability to be executed. Examples of capabilities are *observe*, *grab material*, *transport a injured person*, etc.
- One instance of Trafficability is associated to each edge, representing the edge practical environment for agent moves. Instances of Trafficability could be *Aerial*, *Terrestrial* or more fine-grained properties such as *Forest*, *Field*, *Street*, etc. Note that a trafficability could also be combinations such as *Terrestrial and Aerial*.

2.2. Formalization

We propose here a mathematical formalization of the mission, that can be used as input for the multi-level configuration problem.

A mission is a tuple $m = (V, E, T, TV, C, CT, R, RE)$ where:

- $V = (1, \ldots, n_V)$ is the vector of vertices. We suppose that vertex with number 1 is the base.
- $E = (e_{i,j})_{i,j \in [1..n_V]^2}$ is the adjacency matrix of size n_V^2 that represents the connection between vertices V. For all vertices $i, j \in [1..n_V]^2$, $e_{i,j} = 1$ if there exists an edge between vertices i and j, $e_{i,j} = 0$ otherwise.
- $T = (1, \ldots, n_T)$ is the vector of tasks that have to be performed during the mission.
- $TV = (tv_i)_{i \in [1..n_T]}$ is a vector of size n_T such that for all task $i \in [1..n_T]$, $tv_i \in [1..n_V]$ is the vertex the task i is assigned to.
- $C = (1, \ldots, n_C)$ is the vector of capability types.
- $CT = (ct_i)_{i \in [1..n_T]}$ is a vector of size n_T, such that for each task $i \in [1..n_T]$ in $m, ct_i \in [1..n_C]$ represents the capability required by task i.
- $R = (1, \ldots, n_R)$ is the vector of traficabilities.
- $RE = (re_{i,j})_{i,j \in [1..n_V]^2}$ is a matrix of size n_V^2, such as for each edge $(i, j) \in [1..n_V]^2$, $re_{i,j} \in [1..n_R]$ is the trafficability of the edge $e_{i,j}$ in m.

We call the graph associated to a mission m the pair (V, E). A mission m is said to be well-formed if the following assumptions hold:

- The graph does not contain any edge from a vertex to itself.
$$\forall i \in [1..n_V], e_{i,i} = 0 \tag{1}$$
- The graph is non-oriented and the trafficability matrix is symmetrical.
$$E = E^T \tag{2}$$
$$RE = RE^T \tag{3}$$
- The graph is connected, i.e. from any two vertices i and j, there exists a path of edges connecting them. Formally, $\forall i, j \in [1..n_V]^2$, $\exists k \in \mathbb{N}^*$, $\exists (v_1, \ldots, v_k) \in [1..n_V]^k$, such that:
$$v_1 = i, v_k = j \tag{4}$$
$$\forall r \in [1..k-1], \quad e_{v_r, v_{r+1}} = 1 \tag{5}$$
- For any vertices $i, j \in [1..n_V]^2$, $e_{i,j} = 1$ means that there is exactly one edge between vertices i and j.

In order to illustrate the notations defined previously, we consider the following toy example.

2.3. Toy Problem Mission

We define a simple Search & Rescue mission m, illustrated in Fig. 3, composed of the following elements.

- The vertices vector of locations is $V = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$, where 1 is the "base" ( ), 2 is the "ruins" ( ), and 3 is the "aid camp" ( ).

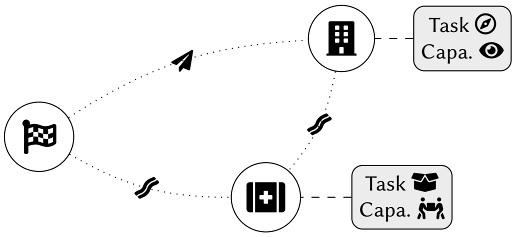


Figure 3: Illustration for Example 2.3. Three locations are considered: a base (🏁), ruins (🏢) to explore, and an aid camp (⛑) to supply. Moving from the base to the ruins requires an aerial agent (✈), while moving to the camp requires a terrestrial agent (🛤).

- The edges matrix of paths is $E = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. For instance, $e_{1,3} = 1$ holds, meaning that it is possible to directly go from vertex 1 ("base") to vertex 3 ("aid camp").
- The tasks vector is $T = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "explore the ruins" (🧭), and 2 is "deliver supplies" (📦).
- The assignment of tasks to the vertices is the vector $TV = \begin{pmatrix} 2 & 3 \end{pmatrix}$, representing that task 1 ("explore the ruins") and task 2 ("deliver supplies") must respectively be executed in vertex 2 ("ruins") and vertex 3 ("aid camp").
- The capabilities vector is $C = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "carry" (🚶), and 2 is "observe" (👁).
- The assignment of capabilities to the tasks is the vector $CT = \begin{pmatrix} 2 & 1 \end{pmatrix}$, meaning that capability 2 ("observe") is required for task 1 ("explore the ruins") and capability 1 ("carry") is required for task 2 ("deliver supplies").
- The traficabilities are $R = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "terrestrial" (🛤), and 2 is "aerial" (✈).
- The assignment of traficabilities to the edges is the matrix $RE = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$. For instance the path $(1, 2)$ has the trafficability 2 ("aerial"), whereas the path $(2, 3)$ has the trafficability 1 ("terrestrial").

3. Agent Configuration Problem

In this section, we present the model associated with the Agent Configuration Problem (ACP), which consists in deciding agents' composition wrt. a catalog of platform types and equipment types, by using the notion of agent pattern.

3.1. Description

As illustrated in Fig. 4, an AgentPattern represents a type of robot or a type of drone that can act somewhat autonomously. Elements composing an agent pattern are divided as follows:

- Platform represents the skeleton of an agent pattern. Each agent pattern has a single platform.
- Each Platform is associated to a unique PlatformType representing the agent pattern skeleton type. Examples of such platform types could be *aerial*, *terrestrial*, *marine*. It would also be possible to consider more fine-grained platform types, such as *quadcopter* or *submarine*. The platform type limits and defines most of the agent pattern characteristics.
- Equipment represents the payload that can equip an agent pattern. An agent pattern can be equipped with several equipments.
- Each Equipment is associated to a unique EquipmentType, which represents the type of the equipment (e.g. camera, sensor, motor).
- Available PlatformTypes and EquipmentTypes are grouped in a Catalog.

An agent is able to interact with the mission throughout two connections to the mission description:

- Each Equipment instance has a set of Capability instances, allowing agents to execute tasks. If an agent pattern is equipped with an equipment that provides the capability associated to a task, then any agent following that pattern will be able to perform the task.
- Each PlatformType instance is associated with a set of Trafficability instances representing the types of environments it is compatible with. Consequently, an agent pattern is compatible with an edge if and only if the edge trafficability belongs to the agent pattern platform type set of compatible traficabilities.

3.2. Formalization

We first formalize the inputs of the agent configuration problem and then define the decision variables. We next present some assumptions on the problems we consider and finally illustrate the concepts on the toy example.

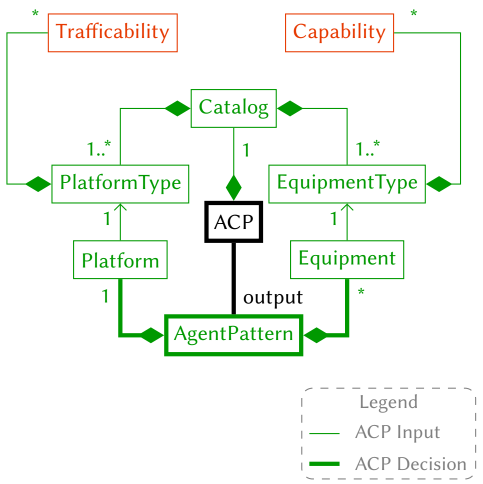


Figure 4: UML representation the ACP.

3.2.1. Inputs

Let m be a mission. A catalog on m is a tuple $cat_m = (P, Q, max_Q, RP, CQ)$ where:

- $P = (1, \ldots, n_P)$ is the platform types vector,
- $Q = (1, \ldots, n_Q)$ is the equipment types vector,
- $max_Q \in \mathbb{N}^*$ is an upper bound on the number of instances of each equipment type that can be carried by an agent pattern,
- $RP = (rp_{i,j})_{i,j \in [1..n_P] \times [1..n_R]}$ is the platform/traficability compatibility matrix of size $n_Q.n_R$. For each platform type $i \in [1..n_P]$ and each trafficability $j \in [1..n_R]$, $rp_{i,j} = 1$ if the platform type i is compatible with trafficability j. Otherwise, $rp_{i,j} = 0$.
- $CQ = (cq_{i,j})_{i,j \in [1..n_Q] \times [1..n_C]}$ is the equipment/capability relation matrix of size $n_Q.n_C$. For each equipment type $i \in [1..n_Q]$ and each capability $j \in [1..n_C]$, $cq_{i,j} = 1$ if the equipment type i provides the capability j. It equals 0 otherwise.

The catalog is the only input of the ACP.

3.2.2. Assumptions

A catalog cat should satisfy the following assumptions.

- **Task Feasibility.** For each task, there is at least one equipment type in the catalog that provides its capability, which translates into:

$$\forall j \in [1..n_T], \sum_{i=1}^{n_Q} cq_{i,ct_j} \geq 1 \tag{6}$$

- **Task Reachability.** For each task, there exists a platform type and a path from the base to the task's vertex such that the platform type is compatible with all the path's edges trafficabilities. Formally, $\forall i \in [1..n_T], \exists j \in [1..n_P], \exists k \in \mathbb{N}^*, (v_1, \ldots, v_k) \in [1..n_V]^k$, *s. t.*

$$v_1 = 1, v_k = tv_i \tag{7}$$

$$\forall r \in [1..k-1]^2, \quad e_{v_r, v_{r+1}} = 1 \tag{8}$$

$$rp_{j, re_{v_r, v_{r+1}}} = 1 \tag{9}$$

Those two assumptions ensure that for each task in the mission, there exists an agent pattern compatible with the task perform it.

3.2.3. Decision Variables

We present here the decision variables that must be assigned a value when solving an ACP. To do so, we first formally define an *agent pattern*.

For a given catalog cat, an agent pattern is a tuple $\mathbf{a}_{cat} = (\mathbf{ap}, \mathbf{AQ})$ where :

- $\mathbf{ap}$ is an integer in $[1..n_P]$ that represents the platform of catalog cat associated with $\mathbf{a}_{cat}$.
- $\mathbf{AQ} = (\mathbf{aq_i})_{i \in [1..n_Q]}$ is the $\mathbf{a}_{cat}$ equipment vector of size n_Q. For all equipment type $i \in [1..n_Q]$, $\mathbf{aq_i}$ is an integer in $[1..max_Q]$ that represents the number of equipment type i present in $\mathbf{a}_{cat}$.

For a given catalog cat, the objective of ACP is to compute a tuple $\mathscr{T}_{cat} = (1, \ldots, n_{\mathscr{T}})$ where each element is an index of an agent pattern, as defined previously, and $n_{\mathscr{T}}$ the number of elements in the tuple.

As we do not consider any constraint in this paper, there are $n_{\mathscr{T}} = n_P \cdot n_Q^{max_Q}$ possible agent patterns. In real world applications, the ACP should of course satisfy some constraints (e.g. max payload, mission's budget, etc.) and could optimize some criteria (e.g. cost minimization). This is out of scope of this paper, and so are the precise definitions of platform and equipment attributes related to them (such as weight, price, etc.). Note that even with constraints consideration, the vector $\mathscr{T}_{cat}$ might be too large to be exhaustively explored.

3.3. Toy Problem ACP

We consider the mission m defined in Subsection 2.3.

We define the catalog cat the following way.

- The platform types vector is $P = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "UAV" () and 2 is "rover" ().
- The equipment types vector is $Q = \begin{pmatrix} 1 & 2 \end{pmatrix}$, where 1 is "camera" () and 2 is "trunk" ().

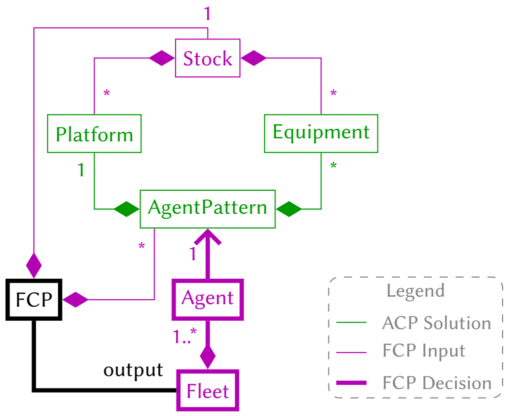


Figure 5: UML representation of the FCP.

- the maximum number for each equipment instance on an agent pattern is $max_Q = 1$.
- The platform/trafficability compatibility matrix is $RP = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. In this example, $rp_{1,2} = 1$ holds, meaning that platform 1 ("UAV") is compatible with the trafficability 2 ("aerial"). However, as $rp_{1,1} = 0$, platform 1 is not compatible trafficability 1 ("terrestrial").
- The equipment/capability relation matrix is $CQ = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. In this example, $rp_{1,1} = 1$ holds, meaning that the equipment 1 ("camera") provides the capability 2 ("observe"). However, $rp_{1,2} = 0$, which means that this equipment does not provide capability 2 ("carry").

The two following agent patterns belong to $\mathcal{T}_{cat}$:

- $\mathbf{a}_1 = (1, \begin{pmatrix} 1 & 0 \end{pmatrix})$ is a UAV equipped with one camera and zero trunk.
- $\mathbf{a}_2 = (2, \begin{pmatrix} 1 & 1 \end{pmatrix})$ is a rover equipped with one camera and one trunk.

There is a total of $n_{\mathcal{T}} = 6$ possible agent patterns ($\mathcal{T}_{cat} = (\mathbf{a}_1, \ldots, \mathbf{a}_6)$).

4. Fleet Configuration Problem

In this section, we present the model associated with the Fleet Configuration Problem (FCP), which aims at deciding the composition of the fleet wrt. the available stock.

4.1. Description

Fig. 5 contains a UML representation of the Fleet Configuration Problem. The FCP class takes as an input the set of AgentPattern computed by the ACP, as presented in the previous section. Its output is a Fleet, *i.e.* a collection of Agents, where each Agent is associated to a unique AgentPattern.

In order to model the fact that equipment and platform are available in limited quantities, we define the class Stock. Such a class is associated to a set of Platforms and a set of Equipments. The FCP takes an instance of Stock as an input. Even if it is clear that the stock will impose hard constraints on FCP, the precise formalization of these constraints is left for future work.

4.2. Formalization

We first formalize the inputs of the agent configuration problem, then define the decision variables and illustrate the formalization on the toy example.

4.2.1. Inputs

Let *cat* be a catalog. The FCP associated to this catalog has two inputs:

- A stock associated with *cat*, denoted s_{cat}, and defined by a pair (P_s, Q_s) where:
  - $P_s = (p_i)_{i \in [1..n_P]}$ is a vector of size n_P such that for each platform type $i \in [1..n_P]$ in *cat*, $p_i \in \mathbb{N}^*$ defines how many type i platform instances are in the stock.
  - $Q_s = (q_j)_{j \in [1..n_Q]}$ is a vector of size n_Q such that for each equipment type $j \in [1..n_Q]$ in *cat*, $q_j \in \mathbb{N}^*$ defines how many type j equipment instances are in the stock.
- A vector of agents pattern $\mathcal{T}_{cat}$. Such a vector can for instance come from the output resulting from the ACP solving.

4.2.2. Decision Variables

Given a catalog *cat*, a stock s_{cat} on this catalog, and $\mathcal{T}_{cat}$ a vector of the agent patterns, a fleet is a tuple $\mathbf{f}_{s_{cat}, \mathcal{T}_{cat}} = (\mathbf{n_a}, \mathscr{A}_\mathbf{f})$ where:

- $\mathbf{n_a}$ is the size of the fleet.
- $\mathscr{A}_\mathbf{f} = (\mathbf{a_i})_{i \in [1..\mathbf{n_a}]}$ is the finite vector of size $\mathbf{n_a}$ of agents in the fleet such as, for each $i \in [1..\mathbf{n_a}]$, $\mathbf{a_i} \in [1..n_{\mathcal{T}}]$ is the index of the agent pattern of the agent i in the fleet.

Note that the model allows to have the same agent pattern present several times in $\mathscr{A}_\mathbf{f}$, representing the fact that there are some identical agents in the fleet.

4.3. Toy Problem FCP

We consider the mission m defined in Subsection 2.3, the catalog cat and the agent patterns $\mathscr{T}_{cat}$ defined in Subsection 3.3.

We define the stock s_{cat} the following way.

- The platform instance vector is $P_s = \begin{pmatrix} 2 & 1 \end{pmatrix}$, meaning that there are 2 instances of type 1 platform ("UAV" -) and 1 instance of type 2 platform ("rover" -) in the stock.
- The equipment instances vector is $Q_s = \begin{pmatrix} 2 & 1 \end{pmatrix}$. In this example, there are two instances of type 1 equipment ("camera" -) and one instance of type 2 equipment ("trunk" -) in the stock.

With this stock, it is possible to configure several fleets of agents. For instance, we define two fleets as follows:

- $\mathbf{f}^1_{s_{cat},\mathscr{T}_{cat}} = (1, (\mathbf{a}_2))$, is a fleet composed of a single agent with the pattern $\mathbf{a}_2$ (a rover equipped with one camera and one trunk - + +).
- $\mathbf{f}^2_{s_{cat},\mathscr{T}_{cat}} = (2, (\mathbf{a}_1, \mathbf{a}_2))$, is a fleet composed of two agents with the respective patterns $\mathbf{a}_1$ (a UAV equipped with one camera - +) and $\mathbf{a}_2$ (a rover equipped with one camera and one trunk - + +).

5. Plan Configuration Problem

In this section, we present the model associated with the Plan Configuration Problem (PCP), which aims at deciding the agents' positions and tasks all along the mission.

5.1. Description

The plan configuration is the last problem to solve in order to get a solution for the multi-level configuration problem. As illustrated on Fig. 6, it takes as input a Mission and a Fleet. Its output is a Plan which consists of an AgentPlan for each Agent in the fleet. For each agent in the fleet, an AgentPlan describes exhaustively at any given time step the position of the agent and the task currently executed, if any.

5.2. Formalization

We first formalize the inputs of the plan configuration problem and then define the decision variables.

5.2.1. Inputs

For a catalog cat, a stock on this catalog, s_{cat}, the PFD associated to this stock requires two additional inputs:

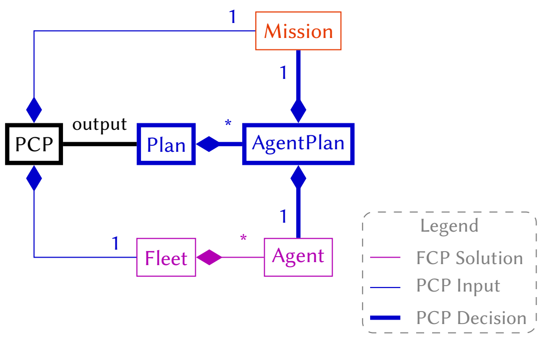


Figure 6: UML representation of the PCP.

- a mission m,
- a fleet $f_{s_{cat},\mathscr{T}_{cat}}$.

5.2.2. Decision Variables

In order to represent the position of each agent in the solution plan, we use binary decision variables (**Vpl** matrix) that indicate whether an agent is at a given position at each time step. Similarly, for each task, we use binary decision variables indicating whether an agent executes this task at the time step (**Tpl** matrix).

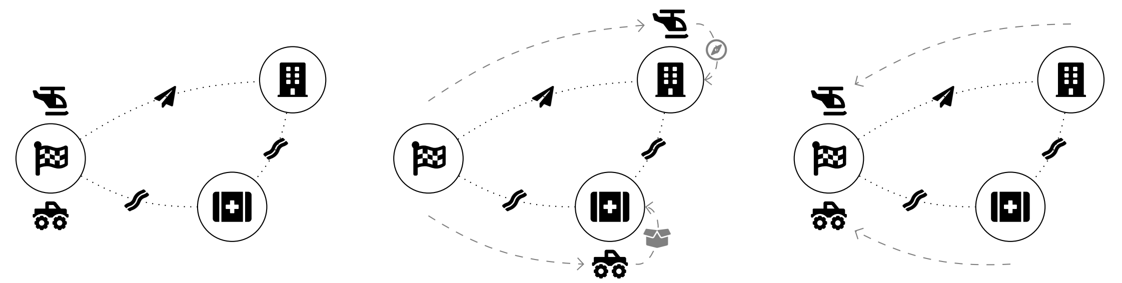
Formally, for a catalog cat, a stock on this catalog, s_{cat}, a mission m, a fleet $f_{s_{cat},\mathscr{T}_{cat}}$, a *plan* is a tuple $\mathbf{pl}_{m,f_{s_{cat},\mathscr{T}_{cat}}} = (\mathbf{H}, \mathbf{Tpl}, \mathbf{Vpl})$ where:

- $\mathbf{H} \in \mathbb{N}^*$ is the temporal plan horizon.
- $\mathbf{Tpl} = (\mathbf{tpl}_{i,j,h})_{i,j,h \in [1..n_a] \times [1..n_T] \times [1..\mathbf{H}]}$ is the allocation of tasks over agents for each time steps, represented as a tensor of size $n_a \cdot n_T \cdot \mathbf{H}$. For each agent $i \in [1..n_a]$, each task $j \in [1..n_T]$ and each time step $h \in [1..\mathbf{H}]$, $\mathbf{tpl}_{i,j,t} = 1$ if the agent $\mathbf{a}_i \in \mathscr{A}_\mathbf{f}$ is executing the task j at the time h. It equals 0 otherwise.
- $\mathbf{Vpl} = (\mathbf{vpl}_{i,j,h})_{i,j,h \in [1..n_a] \times [1..n_V] \times [1..\mathbf{H}]}$ is the position of the agents for each time steps, defined by a tensor of size $n_a \cdot n_T \cdot \mathbf{H}$. For each agent $i \in [1..n_a]$, each task $j \in [1..n_T]$ and each time step $h \in [1..\mathbf{H}]$, $\mathbf{vpl}_{i,j,t}$ equals 1 if the agent $a_i \in \mathscr{A}_\mathbf{f}$ is at the vertex j at the time h. It equals 0 otherwise.

Through this plan formalization, moves of agents are not explicitly described, but this piece of information could be retrieved through their positions.

5.3. Toy problem PCP

We consider the definitions of mission m, catalog cat, introduced in the three previous examples.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 7: Execution of plan $\mathbf{pl}_{m,f_{s_{cat},\mathcal{T}_{cat}}}$ from Example 5.3: starting from the base, the UAV moves to the ruins while the rover moves to the aid camp; then, they both perform the required tasks in their respective locations; finally, they both come back to the base.

We consider the following plan for the fleet $f^2_{s_{cat},\mathcal{T}_{cat}} = (2,(a_1,a_2))$:

$\mathbf{pl}_{m,f_{s_{cat},\mathcal{T}_{cat}}} = (3,\big(\mathbf{Tpl}_1 \quad \mathbf{Tpl}_2\big)),\big(\mathbf{Vpl}_1 \quad \mathbf{Vpl}_2\big))$, illustrated in Fig. 7, where:

- $\mathbf{Tpl}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ is the task allocation matrix of the first agent of the fleet, that has pattern a_1 (platform). It performs the task "explore the ruins" () at time step 2.

- $\mathbf{Vpl}_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ describes the movement of the first agent of the fleet, that has pattern a_1. It starts at the "base" () than goes to the "ruins" () and comes back to the "base" ().

- $\mathbf{Tpl}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ is the task allocation matrix of the second agent of the team, with pattern a_2 (platform). It performs the task "deliver supplies" () at the time step 2.

- $\mathbf{Vpl}_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ describes the movement of the second agent of the team, with pattern a_2. It starts at the "base" () than goes to the "aid camp" () and comes back to the "base" ().

Note that the time steps used in that example plan give a macro view of the agents actions. It would be possible to have a much finer discretization of the time in order to handle temporal constraints such as task duration, or edge traversal duration.

6. Conclusion

In this paper, we model and formalize the multi-level configuration problem for a fleet of heterogeneous agent. This problem is decomposed into three problems, ACP, FCP and PCP and for each of them, we formally define their inputs and their decision variables and we illustrate them on a toy problem. We focus on Search and Rescue missions where tasks have to performed on some nodes of a given graph.

The work presented in this paper is a first step for solving the multi-level configuration problem. As mentioned in the paper, the next step is to formally define the set of constraints and the eventual criteria associated to ACP, FCP and PCP. To do so, it will be possible to study the literature associated with each problem, such as [1] for ACP, [3, 6] for FCP and [7, 8, 9] for PCP.

Then, we have presented the three configuration problems independently but in practice, they are interleaved. For instance the output of ACP is an input of FCP, and the output of FCP is an input for PCP. In the other direction, the evaluation of solutions produced by PCP and FCP can influence the choices made in ACP. If the evaluation of the overall multi-level configuration solution is not satisfactory, there might be several interactions between each level before converging (if any convergence is possible). In order to avoid these interactions, it would be possible to solve all the configuration problems simultaneously. Some works have started contributing towards that objective [10, 11, 12, 2]. Following those works, we aim at proposing a global solver/architecture for solving the multi-level configuration problem.

Finally, we have considered here a simple model of a Search and Rescue mission. It would be possible to make it more realistic in several ways. For instance, it would be possible to consider: more complex mission (e.g. with multiple bases), autonomy constraints on agents forcing them to recharge in some specific locations, more complex tasks (e.g. requiring multiple capabilities, or requiring synchronisation between multiple agents), a non-deterministic setting (e.g. uncertainty on tasks duration) and a dynamic environment (e.g. discover the edges trafficability, agent's loss).

Acknowledgments

The authors would like to thank the ONERA, ISAE-SUPAERO and ENAC Federation for its support of this work. This work is partly founded by the ONERA federative project on cooperative and interactive intelligent multi-robot systems (SICICOD).

References

[1] É. Vareilles, S. Roussel, G. Picard, PERFECT: performant and robust read-to-fly fleet configuration: from robot to mission plan, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proceedings of the 25th International Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 104–107.

[2] C. Lei, W.-H. Lin, L. Miao, A two-stage robust optimization approach for the mobile facility fleet sizing and routing problem under uncertainty, Computers & Operations Research 67 (2016) 75–89.

[3] G. Weiss, Multiagent systems, Second Edition, MIT press, 2013.

[4] G. Radzki, P. Golinska-Dawson, G. Bocewicz, Z. Banaszak, Modelling robust delivery scenarios for a fleet of unmanned aerial vehicles in disaster relief missions, Journal of Intelligent & Robotic Systems 103 (2021) 1–18.

[5] T. Calamoneri, F. Corò, S. Mancini, A realistic model to support rescue operations after an earthquake via uavs, IEEE Access 10 (2022) 6109–6125.

[6] J. F. Hübner, J. S. Sichman, O. Boissier, Moise+ towards a structural, functional, and deontic model for mas organization, in: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, 2002, pp. 501–502.

[7] J. Blythe, An overview of planning under uncertainty, Artificial Intelligence Today: Recent Trends and Developments (2001) 85–110.

[8] Ç. Koç, T. Bektaş, O. Jabali, G. Laporte, Thirty years of heterogeneous vehicle routing, European Journal of Operational Research 249 (2016) 1–21.

[9] L. Berghman, Y. Kergosien, J.-C. Billaut, A review on integrated scheduling and outbound vehicle routing problems, European Journal of Operational Research 311 (2023) 1–23.

[10] R. F. Lemme, E. F. Arruda, L. Bahiense, Optimization model to assess electric vehicles as an alternative for fleet composition in station-based car sharing systems, Transportation Research Part D: Transport and Environment 67 (2019) 173–196.

[11] R. Pinto, A. Lagorio, R. Golini, Urban freight fleet composition problem, IFAC-PapersOnLine 51 (2018) 582–587.

[12] H. R. Sayarshad, R. Tavakkoli-Moghaddam, Solving a multi periodic stochastic model of the rail–car fleet sizing by two-stage optimization formulation, Applied Mathematical Modelling 34 (2010) 1164–1174. doi:https://doi.org/10.1016/j.apm.2009.08.004.

Challenges in Automotive Hardware-Software Co-Configuration

Florian Jost[1,*], Carsten Sinz[2]

[1]Mercedes-Benz AG, Leibnitzstr. 2, Böblingen, 71032, Germany

[2]Karlsruhe University of Applied Sciences, Moltkestraße 30, 76133 Karlsruhe, Germany

Abstract

Car manufacturers offer their customers an enormous number of configuration options to individualize their vehicles. While configuration options mostly covered physical components in the past, over the last years the number of software-related options has increased immensely. Existing systems for car configuration should thus be optimized and extended to handle the shift towards more software-related features, e.g. for automatic driver assistance systems. In this article, we highlight different problems and properties combined systems of hardware-software configurations have to tackle in an automotive context.

Keywords

Automotive Configuration, Hardware-Software Co-Configuration, Future Challenges

1. Introduction

Modern car manufacturers offer a vast number of configuration options for their products. In the past, these options covered parts and functionality that were primarily based on physical components. With the ongoing electrification of cars, the rate of functionality implemented in software and thus also the variance in software is increasing [1].

But not only the number of configuration options is increasing, the same holds for the complexity of their interdependencies. Some options are mutually dependent, others are mutually exclusive. This results in an enormous configuration space with more than 10^{100} possible constructable (valid) configurations for a product line [2]. The inherent complexity of the problem challenges automotive manufactures in all stages of the product lifecycle, from development, through production, sales to after-sales. Due to the increasing importance of software in this context, hardware-software co-configurations are playing an increasingly substantial role.

In this publication, we describe upcoming or already existing problems that arise in the context of increasingly software-driven vehicles.

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.

florian_benedikt.jost@mercedes-benz.com (F. Jost); carsten.sinz@h-ka.de (C. Sinz)

0009-0006-9670-8856 (F. Jost); 0000-0001-9718-1802 (C. Sinz)

2. State of the Art

Classically, the possible configurations through which a car can be realized are represented by configuration options. In addition to different color finishes, these can also describe different engine designs, or optional extras such as an improved infotainment system. An existing order, i.e. a set of configuration options, can then be translated into the physical parts that are needed to manufacture the particular car instance.

Historically, with the increasing emergence of software functionality and the associated ECUs (electronic control units) in the car, the existing hardware configuration systems were adapted to also manage software configurations. But, as automotive software can have a wide variety of requirements for the installed hardware (e.g. sensors for autonomous driving systems), software configuration cannot be treated independent of the hardware configuration. However, this close connection is often not reflected in current configuration systems, where mostly software configuration is treated as a second configuration step, after the physical components have been selected and configured.

Additionally, software often comes with configurable parameters that can be set to different values. As an example, emergency call numbers differ from country to country and have to be set accordingly. The basic functionality of the software remains the same, independent of the value the parameter is set to. Thus, instead of writing software for each possible parameter setting, an ECU runs through an additional configuration step, where correct parameter values are written to the ECU (special bits get set in the programmable ROM of the ECU). The high-level software then can adapt its functionality by reading the corresponding bits in the ROM. In the automotive sector, this configuration step is often called variant coding [3].

Typically, in the automotive industry, configuration options are realized through Boolean variables, so-called *codes*, where each code is represented by a variable name a.k.a. identifier. An option selected by a customer is then reflected by setting the corresponding code to true. Besides codes to register customer's selections, internal codes are used, for example, there can be codes representing spatial regions and codes indicating if the car possesses left or right steering.

The set of valid configurations is described by a set of Boolean formulas (*rules*, *constraints*) that all have to be satisfied in a valid order. Such constraints can express, e.g., mutual exclusivity, as in the selection of left-hand or right-hand steering. But often constraints are much more evolved, encompassing many dozens of codes.

Example:

$$c_1 \to \neg c_2 \land \neg c_3$$

with codes c_1, c_2, c_3. In other words, if we select code c_1, codes c_2 and c_3 cannot be selected.

Handling of parameters is mostly done in separate systems, where valid values (or sometimes even valid combinations of values) are specified via tables listing the admissible settings.

3. Challenges

Since software has become central in modern cars, the question arises whether existing systems, into which the software configuration is mostly just cobbled in, still meet all the necessary or desired requirements.

Until today, the configuration process is still mostly divided into two steps. First the hardware configuration, then – on top of it – the software configuration. However, it is questionable whether this two-step process is still the best approach for existing and future use-cases.

In this section, we present various challenges that prevail in current hardware-centric configuration systems and might require additional consideration in future hardware-software co-configuration systems.

Hardware Upgrade. Automotive manufactures recently started to offer subscription models for features that are not present at the time of sale, but can be retrofitted – often by a simple software switch – into already delivered vehicles. For this, the manufacturers need the possibility to enable functions for vehicles in the field. Typically, this also requires a check, whether the hardware of the car supports the extended functionality.

It might even be the case that an OEM (Original Equipment Manufacturer) allows a retrofit including an update of hardware components. This can occur as follows: In a revision of an existing car series, the hardware is slightly modified and a new software function becomes available. Now, this software function could potentially also be provided in the older model, if the necessary hardware can be retrofitted. Checking whether such an update is possible (and which parts have to be exchanged) requires access to the exact configuration of the delivered car as well as configuration systems that have knowledge about constraints for historic configurations, possibly going back to several years or even decades.

OTA Update. Over-the-air (OTA) software updates present unique challenges for automotive manufacturers. Firstly, the vehicle's software configuration must be fully defined in both hardware and software terms, ensuring that only compatible software satisfying all dependencies is delivered to the vehicle. Additionally, the delivered software must be correctly configured through variant coding, based on the underlying hardware configuration. Therefore, a combined consideration of hardware and software is highly important

Missing Expert. Up to day, software updates are still performed in repair shops by an expert. Manufacturers profit in this context from the expertise of the working staff. Occurring errors can instantly be analyzed by a qualified person. In the best case, the underlying error can be directly fixed by the repair shop staff. Especially in the case of OTA updates, this expertise is missing. In particular, problem analysis and troubleshooting pose a special challenge, as they all have to be done before delivery of the update or – for residual errors – must be fixable remotely, in the worst case by a downgrade to the previous version.

Certification. The automotive sector is highly standardized and regulated. Automotive manufacturers must guarantee that their products satisfy all kinds of standards from different domains. The regulations classically certify a vehicle to satisfy predefined security and safety standards. With the increasing use of driver assistant systems in the automotive sector, the certification requirements, especially in software, grow. In particular, lawmakers want to know in the future exactly which software is delivered in which car. An example is the UN ECE regulation 156,[1] describing the requirements for a software update management system (SUMS) and the future scope for type approval procedures under consideration of the software.

[1] https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update

Frequent Updates. Software updates can be much more frequent than hardware revisions. As software is in an increasing manner not only security but also safety relevant, software updates may have the need to be rolled out quickly. However, as software changes can impact other components of the vehicle, an automatic validity and conformity check of the target vehicle configuration is required. In the best case, after a fix in an existing software package, a package manager should compute a new valid configuration, including the fixed software.

Vehicle function distribution. The distribution of functions in a vehicle and their mapping to ECUs is still done mostly manually. However, with the rapidly changing software architectures in vehicles, the distribution is getting more and more complex. To simplify the initial process in development, algorithmic support is an obvious solution. This requires a detailed specification and documentation of dependencies of the software to be distributed. This initial distribution, we call it *static vehicle function distribution*, can also be extended to the dynamic case, in which functions can be (re-)distributed in real-time to corresponding computing nodes. This allows an improved energy usage, as ECUs can be turned off and on if needed – complex algorithms might even be run in the cloud. For both use cases, a complete description of the hardware and software dependencies is required.

Version Constraints. For software configurations, the specific version of a software package and its dependencies are vital information. In a major software release, dependencies might change drastically, reflecting the changed and extended behavior of the software. However, version constraints are mostly documented in a numeric way, e.g. via *Semantic Versioning*.[2] Whether the currently employed Boolean formalization of constraints is still the best way to address the problem is questionable.

Variance over time. If software updates can still be carried out for older vehicles, new functionalities can also be integrated into existing fleets if technically feasible. Determining which vehicle configurations can still be supplied with new software, as well as documenting and verifying this, is a major challenge given the enormous space of possible configurations. Enabling this variance over larger timeframes will require new mechanisms and considerations.

Software Packaging. In classic computer systems, software configuration problems are often solved with the help of package managers. However, automotive software has additional constraints. In addition to the hardware dependency, there are also complex parameterizations (variant coding) and the need for diagnostic options. This raises the question of how to define software components or packages in order to be able to adopt existing concepts.

[2]https://semver.org

4. Related Work

How to handle hardware/software configurations in the automotive sector has been an active point of research and discussion for several years now [4, 5]. In a survey of German car manufacturers, Sax et al. [6] claim that new ways of checking the consistency of major, regular software updates is an important aspect for not hindering fast development of new functions in the future.

The configuration problem in the automotive industry and solutions to it were already described in the early 2000s. Sinz [7] describes a rule system based on Boolean logic. Here, not only checking individual configurations is covered, but also ways to determine common properties of all valid configurations. Moreover, the mapping from code sets into concrete physical components is also considered. In a later publication, Sinz [8] also describes the verification of such rule systems in order to detect and minimize errors at an early stage. Astesana et al. [9] on the other hand, describe vehicle configurations by using a CSP framework, with Renault as a case study. More recent publications also deal with the topic of classic configurations. Bischoff et al. [10] describes a graphical editor for visualizing and editing item selection rules.

In addition to the control systems mentioned above, there are other approaches to describing variability in general. One of them is feature modeling. In this, the configuration options (features) are often represented in the form of trees, which represent the relations between the features. The analysis of feature models is often performed with the use of SAT solvers [11]. However, analysis approaches using SMT solvers have also been part of recent research [12].

In the area of classic computer systems, configurations are often found in the area of package management and dependency solving. These are mostly so-called component-based systems such as GNU/Linux distributions (e.g. Debian [3]). These contain metadata for software packages, which the package managers utilize in their search for valid configurations. While most package managers initially used ad-hoc solvers [13], nowadays more efficient algorithms from the SAT or CSP community are usually employed [14]. Pinckney et al. [15] recently proposed a package solver, *PacSolve*, for NPM which uses

[3]https://www.debian.org

an SMT approach instead of SAT/CSP solvers. As an alternative to SAT and SMT, there are also publications that describe and solve package update problems with Answer Set Programming [16, 17].

The distribution of different functions in the architecture of a vehicle represents an enormous challenge in modern vehicles. Ruhnau et al. [18] take a first step towards mastering this challenge by describing an ontology for function distribution, covering both static and dynamic distribution.

5. Conclusion

In this paper, we have described various challenges that exist in the area of hardware/software configurations in the automotive sector. Many of these challenges arise from the increasing complexity and relevance of software in vehicles. We have listed that research work already exists for some of the topics. For others, there are promising approaches from related problem areas, the suitability of which we will investigate in more detail in future work.

Acknowledgments

The research presented in this paper was done in the context of the SofDCar (19S21002) project, which is founded by the German Federal Ministry for Economic Affairs and Climate Action.

References

[1] C. Fehling, M. Frank, O. Kopp, Digital sustainability and digital diversification: The two key challenges for automotive software development, in: IEEE 18th Intl. Conf. Software Architecture Companion (ICSA-C), 2021, pp. 162–166.

[2] A. Kübler, C. Zengler, W. Küchlin, Model counting in product configuration, in: Proc. of the 1st Intl. Workshop on Logics for Component Configuration, LoCoCo 2010, Edinburgh, UK, 2010, pp. 44–53.

[3] H. Takimizu, T. Fukaya, Y. Ito, N. Sakano, ECU variant coding system, Mitsubishi Motors Technical Review 18 (2006).

[4] O. Media, Effective hardware-software co-design for automotive systems, 2014. URL: https://embeddedcomputing.com/application/automotive/effective-hardware-software-co-design-for-automotive-systems.

[5] O. Burkacky, J. Deichmann, G. Doll, C. Knochenhauer, Effective hardware-software co-design for automotive systems, 2018. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture.

[6] E. Sax, R. Reussner, H. Guissouma, H. Klare, A Survey on the State and Future of Automotive Software Release and Configuration Management, Technical Report 11, Karlsruher Institut für Technologie (KIT), 2017.

[7] C. Sinz, Baubarkeitsprüfung von Kraftfahrzeugen durch automatisches Beweisen, Diplomarbeit, Universität Tübingen, 1997.

[8] C. Sinz, Verifikation regelbasierter Konfigurationssysteme, Ph.D. thesis, Universität Tübingen, Tübingen, Germany, 2003.

[9] J.-M. Astesana, L. Cosserat, H. Fargier, Constraint-based vehicle configuration: A case study, in: 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, 2010, pp. 68–75.

[10] D. Bischoff, W. Küchlin, O. Kopp, Poseidon: A graphical editor for item selection rules within feature combination rule contexts, in: PLM in Transition Times, Springer, Cham, 2023, pp. 3–14.

[11] D. Benavides, S. Segura, A. Ruiz-Cortés, Automated analysis of feature models 20 years later: A literature review, Information Systems 35 (2010) 615–636.

[12] J. Sprey, C. Sundermann, S. Krieter, M. Nieke, J. Mauro, T. Thüm, I. Schaefer, SMT-based variability analyses in FeatureIDE, in: Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems, ACM, 2020.

[13] P. Abate, R. D. Cosmo, R. Treinen, S. Zacchiroli, A modular package manager architecture, Information and Software Technology 55 (2013) 459–474.

[14] P. Abate, R. D. Cosmo, G. Gousios, S. Zacchiroli, Dependency solving is still hard, but we are getting better at it, in: 27th Intl. Conf. Software Analysis, Evolution and Reengineering (SANER), IEEE, 2020.

[15] D. Pinckney, F. Cassano, A. Guha, J. Bell, M. Culpo, T. Gamblin, Flexible and optimal dependency management via Max-SMT, in: 45th Intl. Conf. Software Engineering (ICSE), 2023, pp. 1418–1429.

[16] M. Gebser, R. Kaminski, T. Schaub, aspcud: A linux package configuration tool based on answer set programming, Electronic Proceedings in Theoretical Computer Science 65 (2011) 12–25.

[17] T. Gamblin, M. Culpo, G. Becker, S. Shudler, Using answer set programming for HPC dependency solving, in: Proc. Intl. Conf. on High Performance Computing, Networking, Storage and Analysis, SC '22, IEEE Press, 2022.

[18] J. Ruhnau, M. Sommer, J. Henle, A. Walz, S. Becker, E. Sax, Ontology for vehicle function distribution, in: IEEE Intl. Systems Conf. (SysCon), Vancouver, Canada, 17-20 April 2023, IEEE, 2023, p. 1–6.

Prospective and retrospective approaches to integrate life cycle assessment in configurators: A multiple case study in the construction industry

Irene Campo Gay[1*], Lars Hvam1[1] and Johan Ernfors[2]

[1] Technical University of Denmark, Anker Engelunds Vej 101 2800 Kongens Lyngby, Denmark
[2] MOKO AB, Humlegårdsgatan 14 114 46 Stockholm, Sweden

Abstract

This study contributes to the evolving dialog on sustainable practices, emphasizing the strategic integration of life cycle assessment (LCA) in configurators to comply with new regulatory standards and achieve environmental objectives. We investigated the application of configurators integrating LCA through a comparative analysis of two case studies in the construction industry: a prospective approach applied during the early design stages, and a retrospective approach using post-design. Our findings illustrate that prospective LCA configurators can significantly influence early design choices and facilitate preliminary environmental impact assessment. Conversely, a retrospective LCA configuration approach offers more precise and accurate assessments based on finalized designs, enabling detailed LCA reporting and saving significant time and effort. The analysis underscores that the application of these approaches is not mutually exclusive. This suggests that a combined strategy could maximize the potential of these tools. Such a combination would facilitate a more dynamic interaction between the early and later design stages, ensuring that the environmental assessment is thorough and iterative. Additionally, it would help the company gain in-depth insights into the environmental aspects of the design process.

Keywords

configurators, construction, environmental impact, life cycle assessment (LCA), sustainability

1. Introduction

Sustainability is widely recognized as a multifaceted concept encompassing three dimensions: environmental, social, and economic. Notably, the environmental dimension plays a foundational role given its direct influence on socioeconomic elements [1]. In assessing environmental impact, particular attention has been given to the environmental impact of products and services. One of the most widespread methodologies for assessing environmental impact is life cycle assessment (LCA) [2].

In this context, the European Commission has highlighted the urgency of making sustainable products the norm across Europe by setting stricter product design and lifecycle standards [3]. As a result, the increasing focus on assessing environmental performance is evidence of the clear need for digital tools to support this process.

Configurators are a widespread technology that emerged in the late 1970s as decision support systems designed to streamline the specification process during product customization [4]. They allow users to select from various options and configurations of a product, automatically adjusting components and features according to user choices. This technology enhances the decision-making process by providing immediate feedback on potential configurations, thereby significantly improving speed, quality, and efficiency [5].

Additionally, configurators enhance efficiency by automating the creation of crucial documents, such as quotes and bills of materials. This automation ensures accuracy and consistency in documentation; it is particularly valuable in complex configurations, where manual processes are prone to errors [6].

The integration of LCA with configurator technology is quite promising for enhancing sustainable product development. By embedding environmental assessment capabilities in configurators, companies can provide real-time data on the environmental impacts of various product options. This integration can facilitate a more informed design, incorporating environmental consequences alongside traditional factors, such as

ConfWS '24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
* Corresponding author.
ircag@dtu.dk (I. Campo Gay)
0000-0002-8962-5386 (I. Campo Gay); 0000-0002-7617-2971 (L. Hvam)

pricing. Furthermore, configurators can enhance the communication of environmental assessment results, a crucial aspect of LCA [7].

Despite the significant potential of this technology, research on the integration of LCA into configurators is still in its early stages. However, over the last five years, this topic has increasingly captured academic interest, as reflected in numerous recent publications [8–19].

Moreover, no research has examined the different implications of LCA in either a prospective or retrospective manner within the context of configurators. This gap in the literature leads to the following research question:

RQ: *What are the implications of a proactive and retrospective of life cycle assessment through configurators?*

We examined two different case studies of companies that have successfully developed configurators with LCA. The first case study involved using this technology during the early design phase to evaluate various design alternatives. The second case study described how a configurator, used over finalized designs, enables precise and accurate LCA. Both case studies pertain to the same sector: the construction industry.

We explored these case studies to assess and compare their impacts, thereby contributing to the research community's understanding of how configurators can be effectively employed to improve environmental development.

The structure of this paper is as follows. In Section 2, we present the theoretical background of LCA typologies in terms of application timing. and we review the academic research conducted on configurators, integrating LCA considerations. In Section 3, we describe the methodology used for analyzing the comparative case studies, and we introduce both case studies. In Section 4, we present the findings from the analysis, and in Section 5, we discuss the implications of the results. Finally, in Section 6, we summarize the key conclusions.

2. Theoretical background

2.1. Prospective and retrospective LCA

The use of LCA is subject to different contexts and can be driven by distinct aims and goals. In terms of the time perspective, LCA can be divided into two primary categories [20, 21]. On the one hand, *retrospective* LCA is aimed at assessing the effects of something that occurred; on the other hand, *prospective* LCA is a forward-looking approach [20, 21].

Retrospective LCA evaluates the environmental impacts of existing products based on actual data. It helps to understand and improve the environmental performance of current technologies [20, 21].

On the other hand, prospective LCA evaluates the potential environmental impacts of products before they are implemented. It is used to guide decision-making during the development phase by predicting future impacts [20, 21].

2.2. Configurators and sustainability

The increasing focus on environmental considerations in the use of configurators has become a significant area of interest over the past five years. This trend is noticeable in the academic community and across various industries. For instance, standard product configuration software applications such as Tacton CPQ are developing their environmental impact assessment capabilities by incorporating LCA features into their applications [22].

Various researchers have also turned their attention to this subject in the academic sector. Given the novelty of the topic, the range of issues discussed in these studies regarding the integration of LCA and configurators is quite diverse, demonstrating the broad scope of the field.

Hankammer et al. [13] extensively reviewed over 900 configurators, providing valuable insights into enhancing sustainability features across sectors. Responding to the need for streamlined LCA assessments, Spreafico et al. [8] introduced I-Tree, a tool that leverages real-time data for efficient eco-assessment. Similarly, Rousseau et al. [10] explored the impact of environmental indicators in configurators, focusing mainly on sustainability enhancement in 3D printing.

To address the nexus between product variety and sustainability, Medini et al. [9] proposed a comprehensive framework, while Wiezorek and Christensen [14] focused on refining configurator architectures to enable better sustainability data communication. In consumer electronics, Hankammer et al. [11] found that default sustainable options significantly influence consumption patterns. Campo Gay et al. [18] analyzed the successful integration of LCA into configurators, guiding users toward sustainable choices. Focusing on sustainability integration, Christensen and Wiezorek [12] aligned configurators with ISO 14040 standards, while Campo Gay and Hvam [17] demonstrated the transformative impact of sustainability-focused configurators, particularly in construction.

Regarding configurators' development, Piroozfar et al. [16] discussed solutions tracking environmental impact, while Helo et al. [15] introduced software streamlining environmental assessments in supply chains. Moreover, Jakobsen et al.'s [19] call to redesign product configuration systems for better sustainability integration tied these efforts together, portraying a concerted push toward deeper sustainability considerations in configuration processes across sectors.

All of these efforts highlight a strong trend toward deepening sustainability considerations within configuration processes.

3. Methods

Given that the advancement of configurators incorporating LCA is still at an early stage, elucidating their full potential and application is a notable challenge. To address this gap, we conducted a qualitative case study analysis comparing two distinct applications of configurator systems within the construction industry. Our objective was to delve deeply into their utilization of LCA and compare their effectiveness to gain in-depth insights.

As highlighted by previous research [23,24], case studies are essential for understanding the key variables, the connections between them, and the reasons behind these relationships.

We identified two case companies using configurators for environmental impact assessment, employing standard LCA methodology. These companies operate within the construction sector in Sweden.

The main reasons are first, that, according to the United Nations Environment Programme (UNEP), the building and construction industry stands as the most polluting industry sector, responsible for 38% of all energy-related CO2 -eq emissions [25]. Consequently, the construction sector has played a pioneering role in shaping standards and regulations, as exemplified by the European standard EN 15804 for environmental product declarations [26], aligned with international LCA methodology standards ISO 14040 and ISO 14044 [7,27].

Second, Sweden has been a leading country in terms of introducing new policies and regulations for the construction sector. Currently, it is compulsory to declare an LCA on new buildings, and beginning in 2025, new projects must adhere to statutory limits on CO_2-eq emission per m2 per year across the life cycle [28].

Consequently, all these factors motivate the construction sector in Sweden to seek out new tools and solutions to support their initiatives and make the studied companies ideal case studies.

3.1. Data gathering

To analyze the first case company, we conducted a series of systematic observations of the configuration process. We evaluated the experiences of the primary configurator implementor involved in the project over a period of four years.

For the company described in the second case, we began with an initial semistructured interview based on the main research question. This was followed by six semistructured interviews to understand the company's working processes and configuration systems. We finalized our analysis with a review of the results by one of the main configurator developers at the company.

3.2. Case company 1

The company is a subsidiary of a large international corporation that operates in Sweden and has approximately 350 employees. It specializes in developing, manufacturing, and marketing cement for infrastructure, such as roads, tunnels, bridges, and residential, commercial, and industrial buildings.

Recognizing the upcoming regulations that will take effect in 2025, which impose limitations on new construction projects, the company saw the need for an early design tool to assist in this process. They developed an LCA configuration to facilitate and promote environmentally friendly design options in the initial stages of projects when decisions are more flexible and have fewer resource implications. This tool assists users in the educational process, encouraging the consideration of less conventional options and more environmentally sustainable solutions.

Given the high level of uncertainty in decision-making during the early design phase of projects, a preliminary LCA was performed. In addition to serving as a decision support tool to address the complexity of environmental and technical requirements, the tool was modeled to quantify LCA to determine the margin of safety concerning maximum statutory limits.

The company has collaborated with external consultants over the past four years to develop this tool, reaching the final testing phase in the first quarter of 2024. Ownership of the tool was transferred to the company during the second quarter of 2024, with full integration into the company's workflow scheduled for completion by June 2024.

3.3. Case company 2

The company is a small enterprise that has been based in Sweden since 2018 and employs 35 people. It specializes in designing and planning the construction aspects of projects. The company uses a configuration system approach to streamline its building design process, which optimizes the overall process.

In response to new regulations requiring LCA declarations for construction projects since 2022, the company has integrated LCA evaluation into its established configurators. To facilitate this, the company uses a commercial solution named *One Click LCA*, a leading cloud-based software solution for

Table 1
Company Case 1: Early Design Stage Configurator Usage (Prospective)

Aspect	Description
Stage of use	Used in the very early design stages for planning
Main purpose of the LCA integration	To make environmentally conscious decisions and compare different solutions
LCA approach	Prospective, integrating LCA from the start of a design
Output	Overview of environmental impacts and technical aspects
Required configuration inputs	Preliminary technical requirements and environmental priorities
LCA integration kind with the configurator	During the configuration.
Impact on the design process	Significant influence over the design approach
Environmental focus	Screening LCA, preliminary impact assessments
Suitability for Projects	New projects with undefined design parameters

creating Environmental Product Declarations (EPDs) and LCA reports for building materials and products.

Consequently, the company has updated its configuration system to automatically generate a comprehensive material list with detailed material quantities in kilograms. These data can then be seamlessly processed by One Click LCA and integrated into the company's database to produce comprehensive EPDs.

4. Results

First, we characterized how each application on the configurator integrating LCA capabilities works and impacts the building design process, focusing on when they are used, what outputs they produce, and their ultimate influence on design decisions and environmental assessments.

Table 1 illustrates the case of Company 1. The application is employed during the early design stage of the building process, representing a prospective approach.

Table 2
Company Case 2: Post-Design Specification of Configurator Usage (Retrospective)

Aspect	Description
Stage of use	Used after the building design is finalized
Main purpose of the LCA integration	To create detailed LCA reports
LCA Approach	Retrospective, LCA applied to finalized designs
Output	Detailed environmental impact reports based on specific materials used and their quantities
Required configuration input	High-level drawing specifications
LCA integration kind with the configurator	After configuration, based on the automatic generation of specifications (i.e., a part list with quantities), the LCA is performed through an external tool (One Click LCA).
Impact on the design process	No or minimal impact on the design.
Environmental Focus	Detailed LCA, focusing on the quantifiable impacts of materials
Suitability for Projects	Projects with set designs needing LCA reflection

In contrast, Table 2 presents the case of Company 2, where the application is used after the design has been finalized, adhering to a retrospective approach.

Subsequently, we examined the implications of choosing either a prospective or a retrospective approach to how a new company's resources, design process, and overall strategy for sustainability are impacted. This should help in understanding the strategic differences between these two approaches.

Table 3 outlines the considerations for a prospective approach, whereas Table 4 details the considerations for a retrospective approach when LCA is integrated into configurators. It is important to note that retrospective design is considered viable only for companies that have already implemented configurators. Implementing a retrospective approach without pre-existing configurators would be significantly costly and inefficient.

Table 3
Considerations for a Prospective Approach to Configurators Integrating LCA

Feature	Evaluation	Explanation
Iterative design	Yes	Facilitates iterative design adjustments during early stages
Approach to design	Prospective	Used to influence initial design choices and integrate LCA
Accuracy	Low	Estimates are broad and based on preliminary data
Influence in design	High	Can significantly alter design outcomes
LCA is the main aim of the configurator	Yes	The primary aim is to guide environmentally conscious design
Further integrations	No	Standalone use for initial design stages
Resource investment in developing the configurator	High	Requires the development of a LCA focused configurator
Cost	Medium/High	Licenses and running cost of configurator tool

Table 4
Considerations for a Retrospective Approach to Configurators Integrating LCA

Feature	Evaluation	Explanation
Iterative design	No	The LCA evaluation occurs once the configuration is finalized
Approach to design	Retrospective	Used after design choices are made
Accuracy	High	Calculations are detailed, based on precise information
Influence in design	None or minimal	The LCA is carried out to reflect on the design rather than modify it
LCA is the main aim of the configurator	No	LCA is secondary and happens thanks to the configuration
Further integrations	Yes	Requires integration with One Click LCA
Resource investment in developing the configurator	Low	Utilizes existing configurator
Cost	Medium/High	While the configurator is in place, there are costs for licenses on external LCA databases.

5. Discussion

A prospective approach in configurators integrating LCA allows for the early detection and mitigation of environmental impacts. In contrast, a retrospective approach ensures that these mitigations are based on precise, real-world data, ultimately enhancing the accuracy and reliability of environmental assessments.

Despite the benefits of a prospective configurator integrating LCA, the higher cost and resource requirements associated with developing and maintaining configurators must be considered. The investment in licenses and running costs of configuration applications can be substantial, but the long-term benefits of reduced environmental impacts and alignment with policies and requirements can compensate for initial costs.

On the other hand, a retrospective approach is highly dependent on completed project design specifications. This approach prevents the flexibility needed to make environmental improvements once the design is finalized. Moreover, while LCA reports on the same product should be comparable and provide precise environmental impact data, the exceptional detail provided by a configuration translates into outstanding detailed LCA reports, which differ from standard LCA reports. For example, this configuration approach could include the consideration of even the smallest details, such as the weight of bolts in a multifamily building construction. Such detailed LCA assessments can result in a misleading comparison between products using the same LCA database, where one report is very detailed (enabled by the configurator), and others are less detailed.

Therefore, the application of prospective and retrospective configurators integrating LCA should not be regarded as mutually exclusive. Incorporating both approaches into a project could substantially streamline efficiency in embracing environmental considerations and reporting LCA. Moreover, such dual applications could enable more dynamic interaction between the early and later design stages, ensuring that the environmental assessment is comprehensive and iterative.

Comparing LCA results from an early design stage with those derived from detailed data collected later in the same project can provide significant insights into product design. This comparison could help companies identify major environmental impact drivers and offer opportunities to reduce environmental assessment uncertainties.

6. Conclusions

The integration of LCA into configurators presents a promising path for enhancing sustainable production practices. We explored the use of configurators integrating LCA at different stages of the design process through two contrasting case studies, a prospective and a retrospective LCA approach in the construction industry.

By employing prospective and retrospective LCA tools, companies can achieve a more thorough understanding of environmental impacts at different project stages, leading to more informed decision-making. This approach not only aids in achieving compliance with evolving regulatory standards but also aligns with broader corporate sustainability goals.

Future research should continue to explore the development and application of these tools across different sectors to fully realize their potential in driving sustainable development.

References

[1] D. Griggs, M. Stafford-Smith, O. Gaffney, J. Rockström, M.C. Öhman, P. Shyamsundar, W. Steffen, G. Glaser, N. Kanie, I. Noble. Policy: Sustainable development goals for people and planet, Nature 495 (2013) 305–307. https://doi.org/10.1038/495305A.

[2] W. Klöpffer, B. Grahl, Life Cycle Assessment (LCA), Wiley Blackwell, Weinheim, Germany, 2014. https://doi.org/10.1002/9783527655625.

[3] European Commission, Communication From the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of The Regions, On Making Sustainable Products the Norm, Brussels, 2022. https://www.europarl.europa.eu/RegData/docs_autres_institutions/commission_europeenne/com/2022/0140/COM_COM(2022)0140_EN.pdf (accessed April 18, 2024).

[4] L.L. Zhang, Product configuration: A review of the state-of-the-art and future research, International Journal of Production Research 52 (2014) 6381–6398. https://doi.org/10.1080/00207543.2014.942012.

[5] A. Felfernig, C. Bagley, J. Tiihonen, L. Wortley, L. Hotz, Benefits of configuration systems, in: Knowledge-Based Configuration: From Research to Business Cases, (Eds. A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen), Elsevier Inc., Waltham, USA, 2014, pp. 29–33. https://doi.org/10.1016/B978-0-12-415817-7.00004-9.

[6] C. Forza, F. Salvador, Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems, Int J Prod Econ 76 (2002) 87–98. https://doi.org/10.1016/S0925-5273(01)00157-8.

[7] ISO, ISO 14044—Environmental management—Life cycle assessment—Requirements and guidelines, 2006. https://sd.ds.dk/Viewer?ProjectNr=M337075P. S. Abril, R. Plant, The patent holder's dilemma: Buy, sell, or troll?, Communications of the ACM 50 (2007) 36–44. doi:10.1145/1188913.1188915.

[8] C. Spreafico, D. Russo, Generating infographics for environmental product declarations (EPDs) with I-tree software, in: Smart Innovation, Systems and Technologies, (Eds. R. J. Howlett & L. C. Jain), Springer Science and Business Media Deutschland GmbH, 2019, pp. 145–154. https://doi.org/10.1007/978-981-13-9271-9_14.

[9] K. Medini, T. Wuest, D. Romero, V. Laforest, Integrating sustainability considerations into product variety and portfolio management, Procedia CIRP 93 (2020) 605–609. https://doi.org/10.1016/J.PROCIR.2020.04.147.

[10] M. Rousseau, K. Medini, D. Romero, T. Wuest, Configurators as a means to leverage customer-centric sustainable systems—Evidence from the 3D-printing domain, in: Procedia CIRP, (Eds. K. Kellens, E. Ferraris, & E. Demeester), Elsevier B.V., 2020: pp. 103–108. https://doi.org/10.1016/j.procir.2021.01.060.

[11] S. Hankammer, R. Kleer, F.T. Piller, Sustainability nudges in the context of customer co-design for consumer electronics, Journal of Business Economics 91 (2021) 897–933. https://doi.org/10.1007/s11573-020-01020-x.

[12] N. Christensen, R. Wiezorek, Enabling mass customization life cycle assessment in product configurators, in: Lecture Notes in Mechanical Engineering , (Eds. A.-L. Andersen, R. Andersen, T. D. Brunoe, M. S. S. Larsen, K. Nielsen, A. Napoleone, S. Kjeldgaard), Springer Science and Business Media Deutschland GmbH, 2022, pp. 819–826. https://doi.org/10.1007/978-3-030-90700-6_93.

[13] S. Hankammer, A.M. Weber, L. Canetta, S.K. Sel, M. Hora, A sustainability based optimization model for starting solutions in toolkits for mass customization, in: 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Ice/Itmc 2017—Proceedings, Institute of Electrical and Electronics Engineers Inc., Madeira, Portugal, 2018, pp. 407–416. https://doi.org/10.1109/ICE.2017.8279914.

[14] R. Wiezorek, N. Christensen, Integrating sustainability information in configurators, in: Proceedings of the 23rd International Configuration Workshop, Vienna, 2021, pp. 65–72.

[15] P. Helo, B. Mayanti, R. Bejarano, C. Sundman, Sustainable supply chains—Managing environmental impact data on product platforms, International Journal of Production Economics 270 (2024). https://doi.org/10.1016/j.ijpe.2024.109160.

[16] P. Piroozfar, S. Shafiee, L. F. Forberg, E. R. P. Farr, Product configurator as a monitoring tool for environmental impact: An AEC perspective, in: Proceedings of the 9th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2023) and the 11th World Mass Customization & Personalization Conference (MCPC2023), Bologna, 2023, pp. 101–109. https://doi.org/10.1007/978-3-031-34821-1_12.

[17] I. Campo Gay, L. Hvam, Sustainability- focused product configurators benefits and expectations: A construction industry case, in: 2023 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2023, Institute of Electrical and Electronics Engineers Inc., Singapore, 2023, pp. 773–777. https://doi.org/10.1109/IEEM58616.2023.10406559.

[18] I. Campo Gay, L. Hvam, A. Haug, Automation of life cycle assessment through configurators, in: 10th International Conference on Mass Customization and Personalization—Community of Europe, Novi Sad, 2022, pp. 19–25.

[19] A. Jakobsen, T. Tambo, M. Kadenic, Greener Information systems for product configuration management: towards adaptation to sustainability requirements, in: Proceedings of the 26th International Conference on Enterprise Information Systems, SCITEPRESS—Science and Technology Publications, Angers, 2024, pp. 100–109. https://doi.org/10.5220/0012737200003690.

[20] A. M. Tillman, Significance of decision-making for LCA methodology, Environmental Impact Assessment Review 20 (2000) 113-123. https://doi.org/10.1016/S0195-9255(99)00035-9.

[21] B.A. Sandén, M. Karlström, Positive and negative feedback in consequential life-cycle assessment, Journal of Cleaner Production 15 (2007) 1469–1481. https://doi.org/10.1016/j.jclepro.2006.03.005.

[22] Environmental Footprint Configuration—Tacton | Tacton, (n.d.). https://www.tacton.com/products/tacton-cpq/environmental-footprint-configuration/ (accessed May 10, 2024).

[23] R.K. Yin, Case Study Research and Applications: Design and Methods, 6th ed., SAGE Publications, Inc., Thousand Oaks, California, 2018.

[24] C. Voss, M. Johnson, J. Godsell, Case research, in: Christer Karlsson (Ed.), Research methods for operations and supply chain management, 3rd Ed., Routledge, London, 2023, pp. 159–189. https://doi.org/10.4324/9781003315001.

[25] UNEP—UN Environment Programme, (n.d.). https://www.unep.org/ (accessed April 14, 2021).
[26] CEN, EN 15804—Sustainability of construction works, Environmental product declarations, Core rules for the product category of construction products, European Committee for Standardization (CEN): Brussels, Belgium (2012).
[27] ISO, ISO 14040—Environmental Performance Evaluation in Engineer-to-Order Companies: An Integrative Framework, 2006.
[28] Swedish National Board of Housing, Building and Planning. (2023, July). Limit values for climate impact from buildings (Report No. 2023:24). Retrieved from https://www.boverket.se/globalassets/engelska/limit-values-for-climate-impact-from-buildings-and-an-expanded-climate-declaration.pdf.

Premises, Challenges and Suggestions for Modelling Building Knowledge using the Configuration Paradigm.

Bart Deschoolmeester[1,*,†], Elise Vareilles[2,3,†]

[1]Lammekensstraat 25 - 2140 Borgerhout, Belgium
[2]ISAE SUPAERO, University of Toulouse, France
[3]IMT Mines Albi, Toulouse University, Albi, France

Abstract
This problem instance paper addresses the need for an industry wide modelling paradigm and language that allows the formalisation and representation of building knowledge by domain experts (architects, engineers). Herein, the special nature of the construction industry (e.g. its openness and semantics) in comparison to other industries and the complexity that arises from this, is recognised. The research needed covers a computation independent meta-model and accompanying modelling language and the added value of the knowledge-based configuration paradigm therein. The research outcome might spark renewed interest in an all-round universal knowledge representation language in the field of building information modelling (BIM) and even prove valuable for other 'less complex' industries.

Keywords
Knowledge Modelling, Building Sector, Configuration, Universal Language

1. Introduction

A modelling environment for the design, construction, operation and end-of-life of buildings, in which it is impossible for the end user to make modelling mistakes because of the integration of personal, company, standardised and regulatory knowledge, has been envisioned since at least 1999 [1]. In addition, the introduction of environmental, social, cost, organisational, etc. objectives would further automate the modelling process through optimisation.

While some attempts have been made in the field of building information modelling, also named BIM, [2][3][4], the quest for a universal knowledge representation language has also been met with scepticism [1][5]: claiming that immediate practical needs should be prioritised or even that this is not (yet) feasible. It can even be argued that the field has adopted a pragmatic approach by focusing on information (as opposed to knowledge) [6], its translation from one environment to another [7], and constraint verification only after modelling [8]. Our proposed research returns to an idealistic view, but finds it promising if based on revised conceptual foundations and the knowledge-based configuration paradigm.

The rest of the paper is as follows. First, in Section 2, the 'open' nature specific to the building industry is presented. In Section 3, the need to call some basic premises of previous efforts into question is addressed. Section 4 introduces the knowledge configuration paradigm and outlines the work of examining the possible benefits and challenges of its application for building knowledge. Lastly, possible further extension of the research is outlined in Section 5.

2. Building Industry as an 'Open' Industry

The need for a universal knowledge representation language (or at least a common meta-model) and the research challenges this provides, arise from the fact that the building industry is possibly the most open industry [1]:

- Many parties are involved in a project and parties change with every project.
- Vast numbers of manufacturers and products for any building part (from traditional to innovative), on any scale (up to the building itself) are available on the market.
- Both a product directly and an onsite composition from products might provide a solution for a required part (e.g. a wall as prefabricated masonry or on site masonry).
- Project specifications often don't prescribe specific products.
- Product delivery might not include some parts but only list its requirements (called 'open systems'

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
†These authors contributed equally.
bartdeschoolmeester@hotmail.com (B. Deschoolmeester); elise.vareilles@imines-albi.fr (E. Vareilles)
https://pagespro.isae-supaero.fr/elise-vareilles/ (E. Vareilles)
0000-0001-6269-8609 (E. Vareilles)

in this text, as opposed to proprietary, 'closed' systems').

This openness is reenforced at a European level through regulation (Construction Product Regulation[9], public procurement[10]) and standardisation (CEN - European Committee for Standardization). This openness entails that most knowledge is generic and generally available in ample building regulations and standards. Designers, contractors and manufacturers refer to these documents and generally only complement them with their specific requirements.

The need for a common language for all the stakeholders is even more acute because of the challenges facing the construction industry: climate and environment, robotics, artificial intelligence, digital twins, etc. and this while facing a shrinking workforce (both engineers and workers).

3. Work Part 1: Basic Premises

In light of the unsuccessful attempts to develop a universal knowledge representation language for the construction sector (see Section 1), it is necessary to first list these experiments, examine their potential shortcomings and generate new ideas and approaches. Based on this work, it will then be possible to define the premises of a meta-model and its accompanying modelling language.

A preliminary examination already allows some underpinnings of previous efforts to be called into question.

Firstly, are existing attempts sufficiently intuitive? The sheer volume of available building expertise will necessitate the creation, verification and maintenance of knowledge models as a collaborative endeavour to be done by domain experts (e.g. architects and engineers) directly without a need for intermediaries like knowledge engineers.

Secondly, are these efforts ontological sufficiently sound? Some examples of overlooked building ontology:

- A building concept can play different semantic roles: it can simultaneous be a conceptual 'container' of parts, items, variants and positions. For instance, a window is composed of parts for its operation: generally, a frame, glazing(s) and hardware. Yet, in a project, the concept might also represent more than one window, for example, a generalisation of the 4 physical windows (items) of the front facade. The concept might also express the variants allowed in the specification (e.g. the designer allows freedom in the choice of hardware to the contractor) or offered by the product (a window available in different heights). Lastly, variability can also exist within a single physical item (called positions in this text): a window can be open or closed, supports for raised office floors having an adjustable height or a ventilation unit with different flow rates. Therefore, at least conceptually, properties must be thought of as potentially having different domains over its parts, items, variants and positions.
- Any level of abstraction should be allowed from the obvious generic concept 'door', over 'partition' (covering window, door, wall, floor, etc.) up to a 'building object' concept.
- Innovative products exist for any building part and therefore must be expected: a generic concept should not be confined to its traditional meaning but allow almost unlimited heterogeneity.
- The semantics of the aforementioned 'position' can be further developed to also hold changes like the onsite length adjustment of a beam, the removal, addition or replacement of a part (e.g. a filter change), or the different installation or use options of a product. With the addition of a 'location' and 'time' property an item could be tracked in space and time, with each change being a new position. Thus covering the complete life-cycle.
- The semantics of the hierarchical relations between a concept and its parts and items respectively, should not be confined to their traditional definitions. A concept is primarily a generalisation of its items but this relation can have a part-like meaning through emergent properties like cardinality, overall cost, energy loss etc. Likewise, a concept might have properties that are a generalisation of the part properties: for example, a masonry wall concept enforces the same colour domain for mortar and bricks.
- The ontology should be polyhierarchical (a single concept occurs in more than in one place) [11]: for example, products exist that act as roof boards and roof insulation or the window grille is simultaneously part of the window and the ventilation system.
- Within the partonomy there is also a need for the idea of 'breakdowns': different ways of breaking down a concept into parts. These ways can be disjunct (variants): for example, the choices for the building structure might be frame-like (e.g. wood or steel) or mass-like (e.g. prefab concrete or masonry). Breakdowns can also be conjunct (within a single variant): a building can be subdivided into its structure and total air volume or into floors (with each floor incorporating part of the structure and air volume). Each breakdown (and its parts) can be needed for the representation of knowledge or user requirements.

Lastly, what is the universe of discourse of the attempts? In any industry, knowledge is interconnected, but in the construction industry, due to its open nature, this is scaled to the entire industry. It might therefore be impossible to effectively isolate a particular aspect in a model while striving for its universal use. The work should therefore outline the contours of what constitutes as building knowledge.

4. Work Part 2: Applying Knowledge-based Configuration

The knowledge-based configuration paradigm defines a configuration model as a set of variables with their domains and with product and user constraints limiting the possible combinations of variable values, and a solution (a configuration) as an assignment of single values to all variables consistent with the constraints (e.g. a valid configuration), as in Chapter 6 of [12]. Knowledge-based configuration is a matured and successful area of artificial intelligence, used and integrated across many industries for more than 40 years, as presented in Chapter 1 of [12]. The configuration paradigm will feel intuitive and familiar for most building professionals: a (product independent) specification as a solution space; a building as a configuration; design choices as constraints; configurable products like drywall systems, roof systems, insulation systems. An intensional, declarative representation through domains and constraints might therefore prove to be a good fit for construction knowledge

Another appealing aspect is the possibility of a representation that is non-causal, meaning that in a particular constraint which variables are input and which are output need not be defined. Though the building modelling process is largely experienced as procedural, directional, top-down, where decisions thought of as the most impactful, like the overall shape of the building, are taken first and then gradually more detailed decisions are taken, it is argued that this must not be imposed by the modelling environment. Light requirements might determine the number and shape of windows instead of the other way around [13], or standard sizes of plywood sheets determine the size of a construction to avoid waste [13]. In light of circularity, products available for reuse might even become requirements instead of solutions. The upcoming practice of early involvement of all stakeholders entails the registering of big and small requirements before designing is started.

The knowledge-based configuration paradigm might even make the typical iterative design process obsolete, creating substantial savings. Though the knowledge-based configuration paradigm seems promising, some challenges to the paradigm can already be identified.

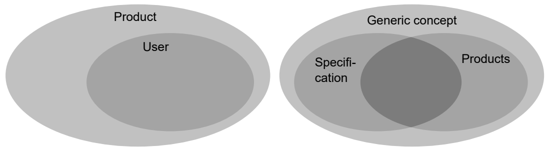


Figure 1: Left: relationship between product and user solution space in a traditional configuration task. Right: relationships between generic concept, project specification and products solutions space in a configuration task for a building project part

Can configuration cover the needs resulting from the work of Section 3: the ontology, the domain of discourse and will it be enough to allow domain experts to take on the role of knowledge engineers? A task resembling the work of [14].

Will the configuration paradigm be able to fully absorb the open character discussed in Section 2?

- The knowledge base will be incomplete. This because of the amount of standards, products, etc. , the gradual nature of the design process or confidentiality (e.g. pricing information). Also, tacit knowledge is prevalent with construction parties.
- As it is impossible for any product knowledge base to contain all building products available on the market, the user requirements (the project specification) do no operate 'within' or on a single product knowledge base, cf. Chapter 6 of [12]. It is rather that both constraints defining multiple products and user requirements operate in the knowledge base of the generic concept (e.g. a generic window, door, wall, etc.) and it is the intersection of the specification and products solution spaces that represents the configurations that provides a solution and this only for the known products (see Fig. 1).
- The user should be presented only with valid options at any one moment in the modelling process. It is therefore not enough to solve for one valid solution but continuously for the complete valid solution space. This is especially necessary in a multi-user environment, where parties operate in each other's solution space.
- Building industry knowledge is distributed. Not only for product knowledge (different manufacturers) but also generic knowledge (building regulations and standards) is generated by different institutions at different geographical levels (municipality, country, EU level, etc.). Expecting all of them to formalise their knowledge on

one location seems unrealistic. The product and generic knowledge base will be distributed and maybe also the project requirements base. Consistency, verification and maintenance of distributed generic knowledge might seem especially challenging.
- A solution is not always a product variant (a single product item). A product item position (a specification might require a specific height for a support, yet a support adjustable in height might be acceptable), a product item part (order the whole product to use only one of its parts) or product items combined (concrete from different suppliers for one single structure or products combined as parts to make up the specified whole) might prove to be equally valid solutions.
- In open systems, as defined in Section 2, the constraints for the not included parts of a (supply side) product might in effect be a product independent (demand side) specification. Making it necessary to solve the product knowledge base first.

5. Further Expansion of Research

Once the conceptual foundation and configuration as a solution established, the research could be extended:

- As touched up in the introduction, a need for optimisation might arise.
- New solving methodologies: computationally more efficient surrogate models might proof to be more practical or the use of generative design where the solution space is explored in an iterative process through single exemplary solutions.
- Propositions for domain expert and end user interface might result from the work.
- New ways of knowledge acquisition like through voluntary open collaboration of domain experts or the use of artificial intelligence (large language models, natural language processing) to extract knowledge.

References

[1] C. M. Eastman, Building product models : computer environments supporting design and construction, 1st. ed., CRC Press LLC, Boca Raton, Florida, 1999.

[2] B.-C. Björk, Requirements and information structures for building product data models, Ph.D. thesis, VTT Building Technology, Espoo, Finland, 1995.

[3] B.-C. Björk, Ratas, a longitudinal case study of an early construction it roadmap project, Journal of Information Technology in Construction 14 (2009) 385–399.

[4] J.-K. Lee, C. M. Eastman, Y. C. Lee, Implementation of a BIM domain-specific language for the building environment rule and analysis, Journal of Intelligent & Robotic Systems 79 (2015) 507–522. doi:10.1007/s10846-014-0117-7.

[5] R. S. Weygant, BIM Content Development. Standards, Strategies, and Best Practices, 1st. ed., Hoboken, NJ, 2011.

[6] F. Liu, A. K. Jallow, C. Anumba, Building knowledge modeling: integrating knowledge in BIM, in: Proceedings of the CIB W78 2013: 30th International Conference – Beijing, w78, 2013, pp. 199–208.

[7] ISO 16739 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries - Part 1: Data scheme, 2nd. ed., International Organization for Standardization, Geneva, Switzerland, 2024.

[8] A. Koutamanis, Planning regulations and modelled constraints in BIM: A dutch case study, Buildings 44 (2024). doi:10.3390/buildings14040939.

[9] Regulation (eu) no 305/2011 of the european parliament and of the council of 9 march 2011 laying down harmonised conditions for the marketing of construction products and repealing council directive 89/106/eec, in: Official Journal of the European Union, 1st. ed., European Union, 2011, pp. 4–43.

[10] Directive 2014/24/eu of the european parliament and of the council of 26 february 2014 on public procurement and repealing directive 2004/18/ec, in: Official Journal of the European Union, volume 57, 1st. ed., European Union, 2014, pp. 65–242. doi:10.3000/19770677.L_2014.094.

[11] ISO 5127 Information and documentation — Foundation and vocabulary, 2nd. ed., International Organization for Standardization, Geneva, Switzerland, 2017.

[12] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1 ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[13] K. M. Kensek, D. E. Noble, Building Information Modeling. BIM in Current and Future Practice, 1st. ed., Hoboken, NJ, 2014.

[14] M. Mohammad-Amini, T. Coudert, É. Vareilles, M. Aldanondo, System Configuration Models: Towards a Specialization Approach, in: 10th IFAC Conference on Manufacturing Modelling, Management and Control, volume 55, Nantes, France, 2022, pp. 1189 – 1194. URL: https://imt-mines-albi.hal.science/hal-03833708. doi:10.1016/j.ifacol.2022.09.551.

Requirements and architectures for green configuration

Richard Comploi-Taupe[1,†], Andreas Falkner[1,*,†], Katrin Müller[2,†] and Sophie Rogenhofer[1,†]

[1]Siemens, Vienna, Austria
[2]Siemens, Berlin, Germany

Abstract
Green Configuration combines product configuration technologies with environmental impact calculations and enables customers to balance cost drivers and environmental impact drivers (such as CO_2 footprint) for their preferred product variants. We analyse requirements for configurable products that go beyond the state of the art of classical Life Cycle Assessment (LCA), and we list corresponding challenges for configurators, such as missing environmental impact data, total costs over the product life cycle, confidence in data accuracy, performance of the calculation, multi-objective optimisation, comparability of the results, and efficient explanations. To address those challenges, we discuss three architecture variants which go beyond sequentially calling separate tools for configuration and LCA: loosely coupled (where the configurator communicates via parameters with the LCA tool), tightly coupled (where the configurator also manages the basic environmental data and lets the LCA tool calculate the impact values for assemblies), and integrated (where the LCA calculation is implemented as part of the configurator). We find that all architectures rely on complete and reliable input data (which might be synthesised offline by data-driven AI methods) and have different advantages and disadvantages concerning efforts for tool vendors, product modellers, and customers.

Keywords
product configuration, sustainability, green configuration

1. Introduction

With the European Green Deal [1, 2], the European Union drives the EU society to a more sustainable future. The EU Agenda 2050 defines environmental, economic, and social goals to be achieved by production systems [3]. Requests for Proposal (RFPs) and other B2B offers of all manufacturing companies will soon require proof of highly sustainable production and operations – due to higher awareness of customers and national authorities, and stricter laws such as the forthcoming Ecodesign for Sustainable Products Regulation (ESPR) [4] or Sustainable Products Initiative (SPI) of the EU.

To persist, companies need to document the Product Carbon Footprint (PCF) or even Product Environmental Footprint (PEF) of all their products transparently and reliably, according to valid or forthcoming regulations like the Digital Product Passport (DPP) [5]. For mass production, processes to assess the environmental impact have already been defined and standardised, e.g., Life Cycle Assessment (LCA) is standardised by ISO 14040 [6].

Product configuration [7] and Industry 4.0 architectures [8] go beyond mass production, and mass customisation allows to manufacture individualised (i.e., lot-size 1) products. The transition towards a circular economy, as required by ESPR, puts challenges to mass customisation and configuration systems, such as the promotion of circularity-based business models, integration of eco-design principles to serve sustainable business demands (i.e., green procurement), and documentation and understanding of the product's material characteristics, manufacturing processes, energy usage, and environmental impacts over the complete life cycle. Only by integrating pre-manufacturing data with data from usage and end-of-life phases can genuine circularity and optimised sustainability (e.g., maximising the product's utility while minimising waste) be reached for configurable products.

The term "Green Configuration" was established a few years ago[1] for this enhancement of configuration tools with environmental impact calculations. This gives the user comprehensive information about the specific effects of their decisions. Small changes in configuration can have a significant impact on the ecological footprint. Multi-objective optimisation strategies make it possible to optimise the product configuration according to desired dimensions (financial and sustainable) depending on specific requirements. Furthermore, provisions must be made so that the final product remains in accordance with the increasingly complex legal framework. This affects not only sales configurators (where customers shall see the expected environmental impact and corresponding costs at the point-of-sale, i.e., before they order a product) but is also vital for engineering configurators (which need to prove that the finally manufactured and deployed product keeps the promises of the sales phase to avoid penalties or non-compliance costs).

Wiezorek and Christensen [11] have given a good overview of the topic, and we will extend their work based on the current developments, e.g., by considering various types of environmental impact (not only CO_2 equivalents) and by integrating the total cost of ownership (TCO) over the complete life cycle (not only the production phase). Our goal is to find alternative architectures for combining configuration and environmental impact calculation and evaluate them w.r.t. user requirements and challenges of their application in practice.

In the next section, we will analyse the state of the art of environmental data and impact calculation. In section 3, we discuss which challenges arise when this is to be applied to configurable products. In section 4, we present the main architectures for green configuration and describe how they deal with those challenges. Finally, we conclude what this can mean for configurator vendors.

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
† These authors contributed equally.
richard.taupe@siemens.com (R. Comploi-Taupe); andreas.a.falkner@siemens.com (A. Falkner); katrin.km.mueller@siemens.com (K. Müller); sophie.rogenhofer@siemens.com (S. Rogenhofer)
0000-0001-7639-1616 (R. Comploi-Taupe); 0000-0002-2894-3284 (A. Falkner)

[1]The term "Green Configuration" has been used more by CPQ solution providers than in academia, e.g., by encoway [9] and CAS [10].

2. Environmental impact assessment: The state of the art

Life Cycle Assessment according to ISO 14040 [6] has been the means of choice for environmental impact assessment for products, processes, and solutions for decades. More and more LCA tools and databases are available, and LCA results are used for Environmental Product Declarations (EPDs) according to ISO 14025 [12]. Examples for commercial providers are SimaPro[2], iPoint[3], and sphera[4]. Ecoinvent[5] is an extensive database used by providers such as SimaPro. Some tools and databases target single environmental indicators only – e.g., SiGREEN [13]. ESTAINIUM[6] is an open network to exchange PCF-related data in a non-profit-oriented way.

The LCAs are based on the product's Bill of Materials (BOM) and Bill of Processes (BOP) along its life cycle. Most LCAs are done after the final product design when the materials and processes are identified. LCA can also be applied earlier in the design process of configurators to improve design decisions before finalisation.

In customer communication, EPDs are often used to show the results of the LCA. However, the EPDs are based on a specific, fully specified product or – less individually and less precisely – on a representative product, an average (fictive) product, or the worst-case product of a homogenous product family. Thus, they cannot help customers decide on product details or with customer-specific optimisation. In the best case, they can give a rough orientation based on existing LCAs for product representatives or typicals.

As EPDs are used for customer communication, Product Category Rules (PCR) and Product Specific Rules (PSR) are defined to provide comparable results [12]. PCRs and PSRs harmonise the system boundaries and provide default parameters for EPDs. However, the usage scenario to be applied in EPD refers to a fictive reference service time. This reference service defines the years of service, load, and operating hours for calculation purposes only. It does not consider the customer-specific usage conditions. Although PCR and PSR aim to provide comparable EPD results within a product category, customers still have to make an effort to relate these results to the individual life cycle conditions and make the right purchase decisions.

Besides the insufficient consideration of the customer-specific usage scenario, the broad range of existing background data sets makes it hard to figure out the product-specific environmental performance, as this is influenced by applied LCA data sets as well. The LCA data sets often provide a market average or a representative example process and do not reflect a specific supplier's product and production-specific environmental impacts. There is still a gap in using primary data along the supplier chain.

Recent initiatives[7] target the PCF accounting and management to improve the primary PCF data share in product accounting and to provide trusted and reliable data along the supply chain. However, even for PCF, several standards and guidelines are in place [14, 15] – and sufficient methods are not yet available to make the data comparable. Large-scale products may require data on millions of materials and components from thousands of suppliers across multiple industrial sectors, which poses considerable challenges to data management and performance.

As PCRs and PSRs try to harmonise the environmental impact assessment within one product category, large-scale systems such as rolling stock, production lines, or process technology are composed of products or assemblies with multiple PCRs and PSRs to be applied, which are not necessarily comparable. Inline environmental assessments are required independently of PCRs and PSRs, especially in large-scale system configuration or turnkey projects. Focusing on customer-specific usage conditions will provide tailored results. However, small changes in the conditions may significantly impact the product's LCA results.

There is little related work concerning combining LCAs with a dynamic modelling approach to consider customer-specific usage or to adapt the background database to future scenarios (cf. Udriot et al. [16] for one example). Such a scenario analyser often applies the same product configuration to multiple usage scenarios. Changes in the product configuration could be made iteratively and sequentially.

Other research reports about work on guidance to integrate LCAs in general and EPDs in particular into configurators and its evaluation in the construction sector [17]. Wiezorek and Christensen [11] suggest an architecture for integrating LCA into a configurator based on a profound analysis of sustainability assessments according to the Ecological Scarcity Method (ESM) and data from the ecoinvent database – focusing on the supply chain and manufacturing phase and mapping all impact to PCF values. A qualitative study [18] lists several advantages that sustainability-focused configurators can potentially provide.

3. Challenges of impact assessment for configurable products

Manufacturing companies need to document not only the PCF but also the PEF (i.e. more environmentally critical substances than just CO2) of all their products transparently and reliably, according to valid or forthcoming regulations like DPP. This must be based on information from suppliers and knowledge about production processes and operations (i.e., usage and end-of-life phases) and includes the selection of suppliers and processes which minimise the overall environmental impact. In addition, economic key performance indicators (KPIs), such as costs for production, transport, usage, disposal, etc. need to be considered and require multi-objective optimisation with good user guidance (including understandable explanations).

The configuration of such an environmentally conscious system is difficult, especially for complex products, because:

- Many suppliers are involved, among them many small and medium-sized enterprises (SMEs), which often cannot provide sufficiently good documentation on materials and PEF (e.g., several thousand suppliers for parts of metro trains).
- Parts have entirely different properties as they come from different industries such as electrical, engineering, or building technology and may interpret environmental KPIs differently.
- Different countries have a wide variety of regulations and certificates (which may even change over

[2] https://simapro.com/
[3] https://www.ipoint-systems.com/
[4] https://sphera.com/
[5] https://ecoinvent.org/
[6] https://www.estainium.eco
[7] Initiatives such as the aforementioned SiGREEN and ESTAINIUM.

time), so different solutions (i.e., combinations of components) are necessary.

- The environmental impact (e.g., concrete PEF values) depends on the production technologies and locations of the suppliers and the location of deployment and conditions at customer sites.
- Sustainability data for many components is missing or questionable, and improvement is difficult as it is out of the control of the system integrator.
- The system configuration is often not yet defined in sufficient detail at the offering time, and therefore, the environmental impacts can only be estimated but not precisely calculated.
- Adaptations during contract negotiations or after deployment can affect compliance and/or performance and require efficient re-calculation and updating of documentation.

To handle those requirements, we need algorithms and techniques for:

- Calculation of all relevant sustainability metrics at point-of-sale: This is not possible in advance (as currently done) because it depends on user decisions, which can lead to billions of potential variants. It must be fast enough to ensure a good user experience and, therefore, requires high performance.
- Reliable aggregation of the values of all sub-parts: This includes highly accurate approximations for missing values specific to the current customer selections. For the usage phase, this cannot be based on sub-parts alone (as is currently done) but on the functionality of the whole product or sub-systems.
- Guided optimisation of several objectives: It is not sufficient to calculate only one (combined, weighted) optimum (as in current tools). The user must be supported in evaluating the Pareto front efficiently and finding the best compromise for conflicting goals.
- Concise visualisation of the results: This helps the user to easily understand the impacts of their decisions. It shall explain the system's confidence in its calculations and where to change a decision to achieve a better result (which goes beyond the capabilities of current systems).

In the remainder of the text, we will focus on the following concrete challenges of Green Configuration:

1. Missing environmental data from suppliers: Many, especially smaller companies, do not yet disclose environmental data for their products (partly because they do not know them themselves). This not only concerns the supply chain, i.e., the impact of the production of those sub-parts, but also their usage and end-of-life processing. To ensure proper LCA calculation, missing data must be synthesised as accurately as possible, i.e., by specific approximations based on machine learning from similar suppliers and/or components, simulation of production and/or operation, using intelligent extrapolation which takes trends into account (e.g., new versions of components typically get better).
2. Unclear impact data for the usage phase: The environmental impact is customer- and even application-specific. It depends on the context, such as operating hours (e.g., whether an engine runs 8 or 24 hours a day) and energy mix (e.g., how much fossil, how much wind power or photovoltaic) [19].
3. Complexity of PEF calculation: The calculation of the complete product's environmental impact (e.g., CO_2 emissions) is more complicated than just adding the corresponding values of all the parts [20]. LCA tools such as Green Digital Twin™ (GDT) [21] or SimaPro implement such details and are certified to comply with the standards.
4. Confidence in calculated data: As the input data come with a certain uncertainty, we must hand over that uncertainty to the intermediate and total values (e.g., with a confidence level or a value range). Plausibility checks (e.g., assembly cannot have less impact than the sum of parts) would be helpful.
5. Multi-objective optimisation: For the customer, it is helpful to know about the impact distribution over the phases (supply chain, production, deployment, usage, end-of-life) and separately for different impact types (energy consumption, pollution, etc.). The corresponding costs (especially TCO) over different expected lifetime periods (e.g., 10 years vs. 20 years) are vital for good decisions. This means the values for all those metrics must be tracked individually.
6. Effective explanations and user guidance: It is insufficient to simply show the user the resulting LCA and TCO values. The user must also understand the causes for those values, i.e., which of their decisions contributed most. Transparency must be established to support users in understanding the impact of a specific configuration on economic and PEF KPIs.
7. Comparability of data: Data often depends on assumptions (such as those mentioned in challenge 2), and players may use different assumptions. To make offers from different vendors comparable, those assumptions and the algorithms used must be disclosed or harmonised, e.g., according to standards such as ISO 14040 [6].

4. Comparison of architectures for green configuration

This section presents architectures with increasing degrees of integration, starting with simply using an existing configurator and feeding its results into an existing or newly customised LCA calculator. In subsections, we will discuss how each deals with the challenges from the previous section and summarise the whole section in a table at the end.

4.1. Status quo: Separate tools

A naïve approach to Green Configuration is sequential – based upon the availability of two separate tools: configurator and LCA calculator. For the configurator (i.e., the left lane in Figure 1), a modeller defines the product model (i.e., variety and dependencies) in a knowledge base (KB) by using the integrated development environment (IDE) for the configurator. A customer or salesperson uses the configurator user interface (UI) to set values to configuration parameters to fulfil their requirements. Continuously, the solver checks compliance with the KB and sets other parameters accordingly. Only when the configuration is finished, the solver hands over the resulting BOM to the LCA tool

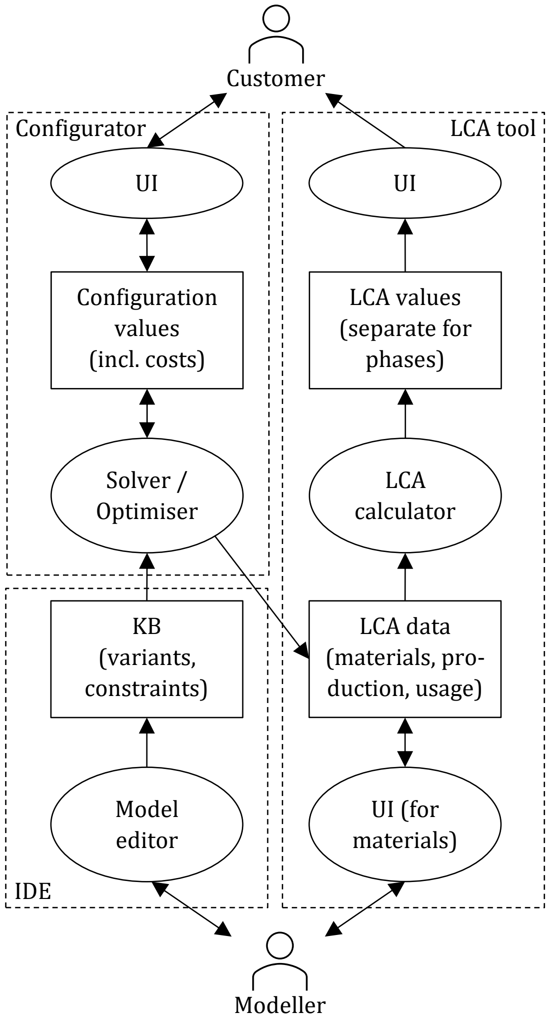


Figure 1: Sequential architecture

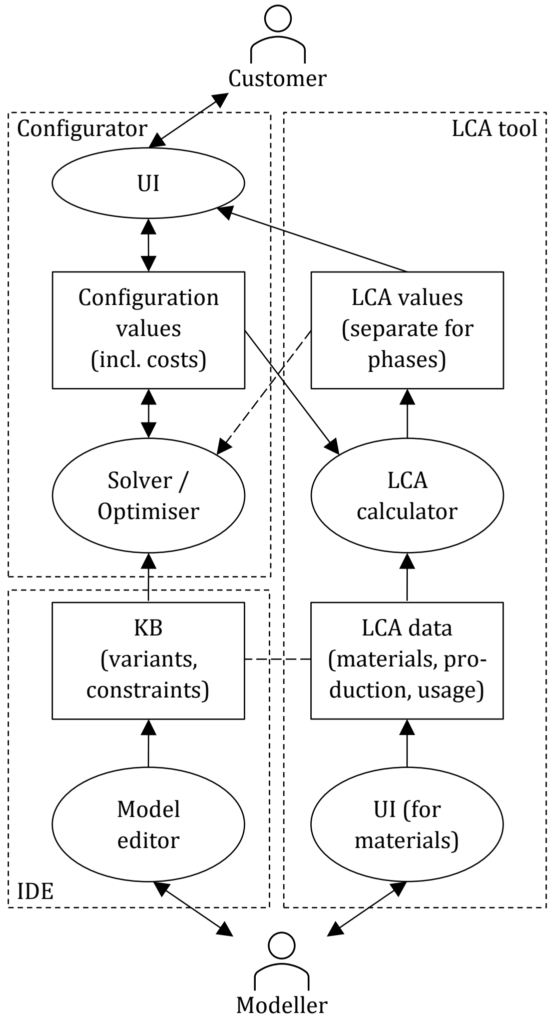


Figure 2: Loosely coupled architecture

(right lane). A (typically other) modeller now collects all necessary LCA data for the materials for all relevant phases (supply chain, production, usage, end-of-life) and calculates the environmental impact values for the product.

We will not go into more detail because this approach does not really combine the two tools and is impractical due to the typically long duration of the manual LCA assessment.

4.2. Loosely coupled architecture

To achieve faster results for the user, one can automate the process. Such a loosely coupled approach was taken by, e.g., Tacton [22]. It is based upon modelling the environmental impact in an LCA tool (such as SimaPro) and synchronising it with the configurator by mapping configuration features with parameters for the LCA (as sketched by the dashed line between KB and LCA data in Figure 2). After each user action in the configurator UI, the LCA calculator is called and returns the adjusted sustainability values to be shown in the configurator UI. The final LCA values may be used for optimisation, i.e., minimisation of environmental impact, in the configurator (indicated by the dashed arrow from the LCA values to the solver).

The main challenge for the configurator vendor is to define a clean generic mapping between configuration and LCA concepts and continuously maintain this interface to comply with the evolving versions of both the configurator and the LCA tool and API. Modellers need much expertise and additional effort because they must specify the LCA model separately from the configurator model and make sure that both are in sync (i.e., define the core structure and the dynamic parameters, include all relevant materials and components, map those included components to configuration features, i.e., parameters). They may even need to involve a tool specialist, at least for the first setup of the system. The configurator users benefit from the proven LCA processes and the typically up-to-date data in the corresponding databases (e.g., ecoinvent). On the other hand, user experience may still be weak because of possibly long response times in interactive use (due to the overhead of calling an external tool and – especially for the first calls – the comparably long time to calculate the resulting value). Optimisation is challenging as the configurator cannot easily access intermediate values for sub-assemblies, thereby steering optimisation in the right direction. This loosely coupled architecture covers the challenges from section 3 in the following way:

1. Missing environmental data from suppliers: Available LCA data for the sub-parts (from suppliers), for the manufacturing tasks (in the own production

process), for various time periods in the operations phase (depending on details of usage and surroundings), and for end-of-life (e.g., recycling efforts) can be reviewed and – if necessary – extended by the modeller in the LCA tool's UI before the configuration process starts. Additionally, an external tool based on machine learning could help to synthesise data offline (this needs to be implemented by other experts).
2. Unclear impact data for usage phase: Information about expected usage can be collected as configuration data and handed over as parameters to the LCA calculator to achieve customer-specific values.
3. Complexity of PEF calculation: The LCA calculator can be trusted to comply with the rules for proper calculation (PCR, PSR).
4. Confidence in calculated data: Current LCA tools do not (yet) sufficiently inform about (missing) accuracy of values.
5. Multi-objective optimisation: LCA values of subparts and sub-assemblies are not available to the optimiser, which can lead to weak (sub-optimal) performance.
6. Effective explanations and user guidance: The configurator UI cannot access the internals of LCA calculation and thus cannot assist the user with explanations and recommendations.
7. Comparability of data: The LCA tool is typically certified. Therefore, the resulting LCA values are comparable to other calculations based on the same standards.

4.3. Tightly coupled architecture

Some LCA tools, e.g., Green Digital Twin™ (GDT) from Siemens, are generic and expect that the LCA data for the LCA calculation is handed over at the call. This can be used for a tightly coupled architecture, where the configurator manages the LCA data and just calls the LCA tool (see Figure 3).

Again, the advantage for the customer is that they are facing just one UI (for configuration and LCA values). But now, the same is true for the modeller (a single UI for configuration and LCA models). This means that the configurator vendor must supply such a modelling UI, which allows the binding of configuration variants to their LCA data (typically extracted from LCA data sets), and a solver which hands the LCA data for the selected variants over to the LCA calculator. The LCA calculator can even be called for parts of the product (not only for the whole product). The tightly coupled approach covers the challenges from section 3 in the following way:

1. Missing environmental data from suppliers: Similarly to the loosely coupled approach, LCA data for the relevant sub-parts can be prepared or synthesised offline.
2. Unclear impact data for usage phase: The configurator hands those LCA data over to the LCA calculator, corresponding to the customer's expected usage.
3. Complexity of PEF calculation: The LCA calculator can be trusted to comply with the rules for proper calculation (PCR, PSR).
4. Confidence in calculated data: Current LCA tools do not (yet) sufficiently inform about (missing) accuracy of values, but as the solver has access to the LCA values of sub-assemblies, it can partly validate them.

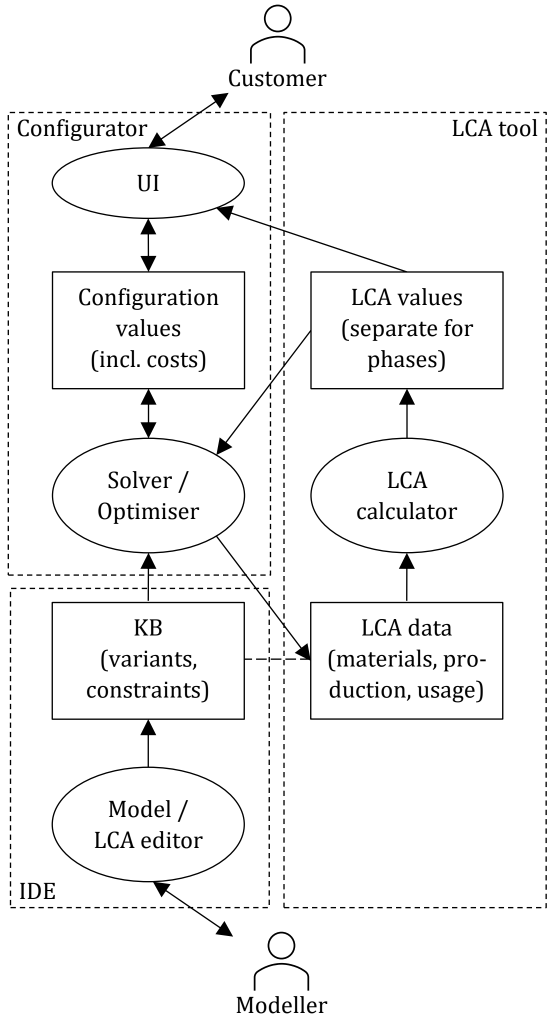


Figure 3: Tightly coupled architecture

5. Multi-objective optimisation: The optimiser can use the LCA values of sub-assemblies for informed heuristics.
6. Effective explanations and user guidance: The configurator UI cannot access the internals of LCA calculation but can use the LCA values of sub-assemblies for some recommendations.
7. Comparability of data: Similar to the loosely coupled approach, the LCA values are comparable to other calculations based on the same standards.

4.4. Integrated architecture

One can go one step further and directly integrate LCA calculation into the configurator by extending the modelling environment (IDE) with a component for LCA and calculating sustainability values directly in the configurator (see Figure 4). Such an approach was taken by, e.g., CAS Merlin [11, 23].

The integrated approach has the advantage that it does not need an explicit mapping to an LCA tool during modelling and can use environmental data during reasoning and optimisation to come up with a more preferred solution. On the other hand, it needs considerable effort for the

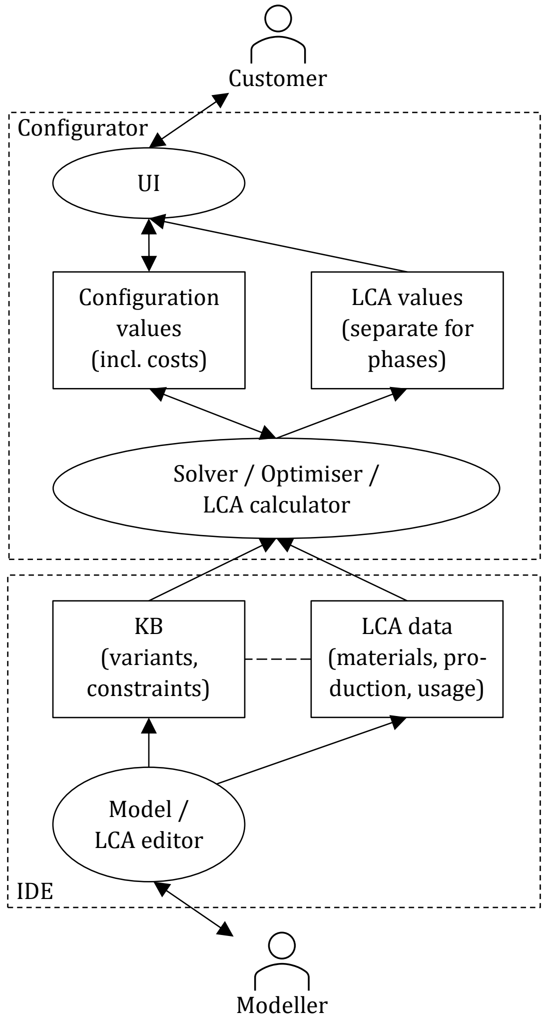


Figure 4: Integrated architecture

configurator vendor to implement the calculation, care for certification (for LCA calculation according to ISO 14040, for EPD generation according to ISO 14025), and continuously maintain it to keep compliance with standards up to date. Development efforts can be reduced if certification is unnecessary, e.g., because customers need not compare their products with competitors but only with their internal variants. The integrated approach covers the challenges from section 3 in the following way:

1. Missing environmental data from suppliers: Similarly to the coupled approaches, LCA data for the relevant sub-parts can be prepared or synthesised offline.
2. Unclear impact data for usage phase: The combined solver and calculator can directly access the expected usage information as specified by the customer to compute the LCA values.
3. Complexity of PEF calculation: Simple impact calculations (e.g., the addition of upstream) can be easily integrated into the solver. Covering the same functionality as an LCA tool and achieving certification requires much more effort by the configurator vendor.
4. Confidence in calculated data: The combined solver and calculator can keep track of the accuracy of the calculated LCA values for assemblies if the accuracy of the input data is known or can be estimated.
5. Multi-objective optimisation: As the optimiser and LCA calculator are fully integrated, intermediate LCA values can efficiently control optimisation.
6. Effective explanations and user guidance: The complete integration of the solver and LCA calculator and full access to all their intermediate data allows for detailed explanations and recommendations.
7. Comparability of data: The extension of the solver with LCA calculation leads to highly individualised LCA values. If the configurator vendor does not achieve certification (e.g., due to high costs and/or efforts), the LCA values may not be comparable to commercial LCA tools.

4.5. Summary

Summing up, all three approaches have strengths and weaknesses when dealing with the challenges. Challenge 1 (missing data) is not discriminating, and the best way to cover it is by extending and/or improving input data offline, e.g. with the help of data-driven AI. Therefore, we rate only challenges 2 to 7 in Table 1 and use a three-valued scale – the approach has strengths, is neutral, or has weaknesses – to condense the arguments from the preceding subsections.

Table 1
Concerning the challenges, the architectures have strengths (+), are neutral (o), or have weaknesses (-)

Challenge	Loosely coupled	Tightly coupled	Integrated
2 - usage phase	o	+	+
3 - calculation	+	+	o
4 - confidence	-	o	+
5 - optimisation	-	o	+
6 - explanations	-	o	+
7 - comparability	+	+	o

The integrated approach offers more value to the customers, e.g. more optimisation possibilities and better explanations. On the other hand, this requires more effort for the configurator developer because they must implement LCA calculations (not just call existing tools or libraries) and care for the necessary certification to make the calculations transparent and comparable.

The coupled approaches take advantage of re-using off-the-shelf LCA calculators and can even hand over configuration information as parameters, but neither (especially the loosely coupled architecture) can easily integrate the calculation results into their reasoning (e.g. for optimisation and explanations). The tightly coupled architecture can access values from sub-assemblies to achieve better usability.

A product modeller may prefer the tightly coupled approach and especially the integrated approach because data management can be done with only one tool: the configurator.

5. Conclusions

Green Configuration, the combination of product configuration technologies with environmental impact calculations, is a vital approach to address sustainability challenges. We have analysed requirements and challenges and discussed several architectures for configurators implementing a green configuration approach.

We have seen that the different architectures have different strengths and weaknesses, advantages and disadvantages. All of them are feasible and require different efforts from stakeholders, i.e., tool vendors, product modellers, and customers. From the viewpoint of a product owner, the selection of their individually preferred architecture depends on the product‘s complexity, the level of product customisation, the number of offers per year, the LCA impact of the usage phase, and the need to enhance customer experience and operational efficiency.

There is much room for future research on efficiently merging sustainability management with configuration lifecycle management, e.g., reference architectures, reliable data exchange, individualised impact calculation, multi-objective optimisation, elaborate standards, etc.

As one of the most important, we see the monetary assessment of PEF as a means of providing an estimate of the TCO. Visualising the monetary impact of configuration decisions over the whole lifecycle of the product will create a real incentive for the customer to choose the more sustainable product configuration (e.g., less energy costs during the operation phase). Green Configuration extended with TCO minimisation can lead to a triple-win situation: minimised total cost of ownership for the customer, increased demand for high-quality products for the industry, and less environmental damage.

Green Configuration enables the creation and scale of application-specific EPDs and DPPs based on more precise information and assumptions on the concrete product properties and usage. To make such specifically customised values comparable between tools, existing standards like ISO 14040 and the ISO 14020 series [24] need to be adapted or extended. Transparency of the individual impact values per phase and/or criterion is necessary for well-founded decisions.

Disclaimer: Much of the presented work is “thought work”. Currently, we are working on prototypes to confirm the ideas and results in practice.

References

[1] European Commission, The European Green Deal: Striving to be the first climate-neutral continent, 2020. URL: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en, accessed 2024-07-19.

[2] European Commission, The European Green Deal, 2019. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM%3A2019%3A640%3AFIN, accessed 2024-07-19.

[3] S. Muench, E. Stoermer, K. Jensen, T. Asikainen, M. Salvi, F. Scapolo, Towards a green & digital future, 2022. doi:10.2760/54.

[4] European Commission, Proposal for ecodesign for sustainable products regulation, 2022. URL: https://environment.ec.europa.eu/publications/proposal-ecodesign-sustainable-products-regulation_en, accessed 2024-07-19.

[5] T. Götz, H. Berg, M. Jansen, T. Adisorn, D. Cembrero, S. Markkanen, T. Chowdhury, Digital Product Passport: The ticket to achieving a climate neutral and circular european economy?, 2022. URL: https://circulareconomy.europa.eu/platform/en/knowledge/digital-product-passport-ticket-achieving-climate-neutral-and-circular-european-economy, accessed 2024-07-19.

[6] International Organization for Standardization, ISO 14040:2006 Environmental management — Life cycle assessment — Principles and framework, 2006.

[7] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), Knowledge-based Configuration: From Research to Business Cases, 1st ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[8] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Business & Information Systems Engineering 6 (2014) 239–242. doi:10.1007/s12599-014-0334-4.

[9] S. Keinitz, Corporate sustainability – how Green Configuration can help!, 2023. URL: https://www.encoway.de/en/blog/green-configuration/, accessed 2024-07-19.

[10] R. Wiezorek, CPQ-Software: Green Configuration für mehr Klimaschutz, 2022. URL: https://www.digital-engineering-magazin.de/cpq-software-green-configuration-fuer-mehr-klimaschutz-a-dc586d2681e3701d3088e0a405ee6185/, accessed 2024-07-19.

[11] R. Wiezorek, N. Christensen, Integrating sustainability information in configurators, in: M. Aldanondo, A. A. Falkner, A. Felfernig, M. Stettinger (Eds.), 23rd International Configuration Workshop, volume 2945 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 65–72. URL: https://ceur-ws.org/Vol-2945/52-RW-ConfWS21_paper_16.pdf.

[12] International Organization for Standardization, ISO 14025:2006 Environmental labels and declarations — Type III environmental declarations — Principles and procedures, 2006.

[13] Siemens, Decarbonization starts with data, 2024. URL: https://www.siemens.com/global/en/company/sustainability/product-carbon-footprint.html, accessed 2024-07-19.

[14] World Business Council for Sustainable Development, Pathfinder framework – guidance for the accounting and exchange of product life cycle emissions, 2021. URL: https://www.wbcsd.org/resources/guidance-for-the-accounting-and-exchange-of-product-life-cycle-emissions/, accessed 2024-07-19.

[15] Together for Sustainability, The product carbon footprint guideline for the chemical industry, 2024. URL: https://www.tfs-initiative.com/app/uploads/2024/03/TfS_PCF_guidelines_2024_EN_pages-low.pdf, accessed 2024-07-19.

[16] M. Udriot, K. Treyer, O. Buhler, L. Etesi, E. David, V. Girardin, Rapid life cycle assessment software for future space transportation vehicles design, in: Aerospace Europe Conference 2023, 2023. doi:10.13009/EUCASS2023-015.

[17] I. Campo Gay, L. Hvam, A. Haug, Automation of life cycle assessment through configurators, in: Z. Anišić, C. Forza (Eds.), 10th International Conference on Mass Customization and Personalization, 2022, pp. 19–25. URL: https://mcp-ce.org/wp-content/uploads/2022/10/5.pdf.

[18] I. Campo Gay, L. Hvam, Sustainability-focused product configurators benefits and expectations: A

construction industry case, in: IEEE International Conference on Industrial Engineering and Engineering Management, IEEE, 2023. doi:10.1109/ieem58616.2023.10406559.
[19] H. Ritchie, P. Rosado, Energy mix, Our World in Data (2020). URL: https://ourworldindata.org/energy-mix.
[20] European Commission, Product Environmental Footprint Category Rules Guidance, version 6.3, 2018. URL: https://eplca.jrc.ec.europa.eu/permalink/PEFCR_guidance_v6.3-2.pdf, accessed 2024-07-19.
[21] Siemens, Finding the right balance between costs and the carbon footprint, 2024. URL: https://resources.sw.siemens.com/en-US/case-study-siemensag, accessed 2024-07-26.
[22] Tacton Systems AB, Tacton CPQ environmental footprint configuration, 2024. URL: https://www.tacton.com/products/tacton-cpq/environmental-footprint-configuration/, accessed 2024-07-19.
[23] CAS Software AG, GreenConfiguration mit CAS Merlin CPQ, 2024. URL: https://www.cas.de/infothek/wissenswertes/cpq-wissen/green-configuration/, accessed 2024-07-19.
[24] International Organization for Standardization, ISO 14020:2022 Environmental statements and programmes for products — Principles and general requirements, 2022.

Developing an Algorithm Selector for Green Configuration in Scheduling Problems

Carlos March[1,*,†], Christian Pérez[1,2,†] and Miguel Salido[1]

[1]Universitat Politècnica de València, Instituto de Automática e Informática Industrial, Camì de Vera S/N, Valencia, Spain
[2]Universitat Politècnica de València, valgrAI - Valencian Graduate School and Research Network of Artificial Intelligence, Camì de Vera S/N, Valencia, Spain

Abstract

The Job Shop Scheduling Problem (JSP) is central to operations research, primarily optimizing energy efficiency due to its profound environmental and economic implications. Efficient scheduling enhances production metrics and mitigates energy consumption, thus effectively balancing productivity and sustainability objectives. Given the intricate and diverse nature of JSP instances, along with the array of algorithms developed to tackle these challenges, an intelligent algorithm selection tool becomes paramount. This paper introduces a framework designed to identify key problem features that characterize its complexity and guide the selection of suitable algorithms. Leveraging machine learning techniques, particularly XGBoost, the framework recommends optimal solvers such as GUROBI, CPLEX, and GECODE for efficient JSP scheduling. GUROBI excels with smaller instances, while GECODE demonstrates robust scalability for complex scenarios. The proposed algorithm selector achieves an accuracy of 84.51% in recommending the best algorithm for solving new JSP instances, highlighting its efficacy in algorithm selection. By refining feature extraction methodologies, the framework aims to broaden its applicability across diverse JSP scenarios, thereby advancing efficiency and sustainability in manufacturing logistics.

Keywords

Job Shop Scheduling Problem, Energy Efficiency, Algorithm Selection, Machine Learning, Feature Extraction

1. Introduction

The Job Shop Scheduling Problem (JSP) is a cornerstone issue in operations research and optimization, serving as a critical benchmark for assessing the performance of various algorithms. JSP entails the complex task of scheduling jobs on machines in a manufacturing environment to optimize several performance metrics, such as makespan, flow time, tardiness, resource utilization, and energy consumption [1]. Effective benchmarking of JSP solutions requires a multi-faceted evaluation of these metrics, particularly focusing on makespan, energy consumption, and tardiness to gauge scheduling efficiency and resource utilization [2]. Tools like JSPLIB play a vital role in these benchmarking efforts by providing researchers with diverse instances derived from significant studies and experiments, thereby enhancing the evaluation of algorithms [3].

Understanding the characteristics of problem instances is essential for effective benchmarking in JSP. Critical factors include the number of jobs and machines, variability in processing times, machine availability, and precedence relationships, all of which significantly impact algorithm performance [4]. Additionally, considering energy consumption, which varies based on machine speed and operational factors, adds another layer of complexity [5]. Achieving a balance between energy consumption and scheduling decisions is crucial for attaining energy efficiency without compromising production goals [6].

JSP's focus on energy efficiency has intensified in recent years due to its substantial environmental and economic impacts [7]. Researchers have investigated strategies such as employing speed-adjustable machines and vehicles to minimize energy consumption while maintaining productivity [8]. Advanced algorithms and optimization techniques have been developed to address these energy-related challenges, taking into account factors like machine speed, idle time, and energy requirements [9]. Real-world implementations of these strategies have demonstrated tangible benefits, including cost savings and positive environmental effects [10].

In addition to traditional optimization methods, machine learning techniques are increasingly being utilized to recommend algorithms for solving problems within the JSP family. For instance, Müller et al. designed a system capable of selecting the most suitable solver for addressing Flexible JSP by leveraging machine learning approaches [11]. Similarly, Strassl and Musliu [12] analyzed JSP instances without energy consumption from the literature to extract features that inform algorithm performance, resulting in a homogeneous set of instances with consistent characteristics. These features were then used to train various models, with Random Forest achieving the highest accuracy at 90% [12].

In conclusion, the integration of machine learning techniques into JSP research provides new avenues for improving algorithm selection and performance, particularly in handling complex and varied instances. This integration enhances the efficiency and effectiveness of job shop scheduling by combining the strengths of traditional optimization approaches with innovative machine learning methods. The ongoing advancements in this field are driving both academic research and practical applications toward more sustainable and innovative solutions.

2. Problem Description and Model Formulation

The JSP tackled in this study emphasizes its intricate energy considerations. The JSP poses a significant computational challenge, being NP-complete due to its difficulty finding optimal solutions within reasonable time frames.

The core challenge of the JSP involves optimizing task allocation across multiple jobs and machines while minimizing key criteria, notably the total job completion time. How-

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
†These authors contributed equally.
cmarmoy@upv.es (C. March); cripeber@upv.es (C. Pérez); msalido@upv.es (M. Salido)
https://gps.blogs.upv.es/ (C. March); https://gps.blogs.upv.es/ (C. Pérez); https://gps.blogs.upv.es/ (M. Salido)
0009-0009-7525-9133 (C. March); 0000-0002-9121-7939 (C. Pérez); 0000-0002-4835-4057 (M. Salido)

ever, achieving this optimization is complex due to various real-world constraints and dependencies, contributing to the JSP's NP-completeness. The combinatorial explosion of possible job and machine combinations further complicates the problem, making exhaustive exploration impractical as the number of jobs and machines grows.

2.1. Mixed Integer Programming

The JSP involves various sets, parameters, variables, and constraints crucial for formulation and solution:

Sets:

- $J = \{1, \ldots, n\}$, the set of jobs.
- $M = \{1, \ldots, m\}$, the set of machines.
- $S = \{1, \ldots, s\}$, the set of speeds.
- $T_j, \forall j \in J$, the set of tasks in job j. In standard JSP $T_j = M$.

Parameters:

- $D_{jt}, \forall j \in J, \forall t \in T_j$, the due date of task job t_{jt}.
- $R_{jt}, \forall j \in J, \forall t \in T_j$, the release date of task job t_{jt}.
- $P_{jts}, \forall j \in J, \forall t \in T_j$, the processing time of task job t_{jt} on machine t with speed s.
- $E_{jts}, \forall j \in J, \forall t \in T_j$, the energy consumption for processing task job t_{jt} with speed s.

Variables:

- $c_{jt}, \forall j \in J, \forall t \in T_j$, the completion time of task job t_{jt}
- $tt_{jt}, \forall j \in J, \forall t \in T_j$, tardiness of task job t_{jt} with respect to its due date
- $x_{mjts} \in \{0, 1\}, \forall m \in M, \forall j \in J, t \in T_j$, binary sequencing variables (i.e., $x_{mjts} = 1$ denotes that task t of job j is performed with speed s on machine m)
- $y_{mijpq} \in \{0, 1\}, \forall m \in M, \forall i, j \in J, \forall p, q \in T_i, T_j, i \neq j$, binary assignment variables (i.e., $y_{ijpq} = 1$ denotes that task p of job i precedes task q of job j on machine m)

$$\phi^* = \underset{\phi \in \Phi}{\arg\min}\, [MK(\phi), EC(\phi), TT(\phi)] \tag{1}$$

subject to:

$$\sum_{m \in M} x_{mjts} = 1 \tag{2}$$
$$\forall j \in J,\ \forall t \in T_j\ \forall s \in S$$

$$\sum_{m \in M} y_{mijpq} = 1 \tag{3}$$
$$\forall i, j \in J,\ \forall p, q \in T_i, T_j,\ i \neq j, p \leq q$$

$$tt_{mjt} \geq c_{mjt} - D_{jt} \tag{4}$$
$$\forall m \in M, \forall j \in J,\ \forall t \in T_j,\ x_{mjt} = 1$$

$$c_{mjt} \geq R_{jt} + P_{mjts} \tag{5}$$
$$\forall m \in M,\ \forall j \in J,\ \forall t \in T_j,\ \forall s \in S,\ x_{mjts} = 1$$

$$c_{mjs} \geq c_{mip} + P_{mips} \tag{6}$$
$$\forall m \in M,\ \forall i, j \in J,\ \forall p, q \in T_i, T_j,\ \forall s \in S,\ i \neq j\ \wedge\ p < q\ \wedge y_{mijps} = 1$$

$$c_{mjt} \geq 0\ , t_{mjt} \geq 0 \tag{7}$$
$$\forall m \in M, \forall j \in J \forall t \in T_j$$

This model seeks the optimal solution ϕ^* that minimizes the three measures mentioned in equation 1. considering the constraints associated: the maximum makespan of all task jobs $MK(\phi)$, the total energy consumption $EC(\phi)$, and the total tardiness $TT(\phi)$. The simultaneous optimization of these objectives requires a delicate balance between the various considerations and constraints of the problem. Therefore, two approaches to optimizing the problem solutions are proposed, allowing us to analyze the methods' behavior better.

2.2. Mono-objective optimization

This section presents the mono-objective optimization for a specific scheduling problem involving multiple jobs and machines, emphasizing key performance metrics such as makespan, energy consumption, and total tardiness.

$$f^m = \max_{\substack{j \in J \\ m \in M}} (c_{jm}) \tag{8}$$

$$f^e = \sum_{j \in J} \sum_{t \in T_j} E_{jt} \tag{9}$$

$$f^{tt} = \sum_{m \in M} \sum_{j \in J} \sum_{t \in T_j} tt_{mjt} \tag{10}$$

Equation 8 represents the makespan, which is the maximum completion time among all machines, by calculating the total processing time of all job tasks on each machine and selecting the maximum value across all machines. Equation 9 describes energy consumption by computing the total energy consumed by all job tasks across all machines. Lastly, Equation 10 is formulated to show the total tardiness, which represents the number of time units of each job or operation that are performed outside its time window, i.e., the period of time between the release date and the due date.

$$min\ \frac{f^m - m_1^-}{m_1^+ - m_1^-} + \frac{f^e - m_2^-}{m_2^+ - m_2^-} + \frac{f^{tt}}{m_1^+} \tag{11}$$

Minimizing the objective Function 11 aims to find a solution that achieves a balanced trade-off among the components. The values $m_{1,2}^+$ and $m_{1,2}^-$ are used to normalize the ϕ^* solution obtained in the three-dimensional objective space. This allows a correct comparison between the values of the objective function in minimizing the problem, giving the same weight to all the parts, and avoiding any of the variables dominating the search.

$$m_1^+ = \sum_{j \in J} (\sum_{m \in M} \max_{s \in S}(P_{jms})) \tag{12}$$

$$m_2^+ = \sum_{m \in M} (\sum_{j \in J} \max_{s \in S}(E_{jms})) \quad (13)$$

$$m_1^- = \max_{j \in J}(\sum_{m \in M} \min_{s \in S}(P_{jms})) \quad (14)$$

$$m_2^- = \sum_{m \in M} (\sum_{j \in J} \min_{s \in S}(E_{jms})) \quad (15)$$

Equations between 12 and 15 determine both maximum (m_1^+ and m_2^+) and minimum (m_1^- and m_2^-) values for makespan and energy consumption respectively. For makespan, these values signify the maximum and minimum completion times across all machines, accounting for the maximum and minimum processing times of job tasks on each machine. Similarly, in terms of energy consumption, they represent the maximum and minimum energy utilized among all machines, considering the maximum and minimum energy consumption of all job tasks on each machine.

3. Algorithm Selector

The selection of algorithms for a given problem JSP involves identifying the most appropriate algorithm from a collection capable of solving JSP, taking into account the specific characteristics of JSP. Rubinoff [13] formalized this process of algorithm selection. Rubinoff defined key elements, including the problem space X, representing all instances of JSP; the algorithm space A, encompassing algorithms capable of solving any $jsp \in X$; and a performance metric y, which quantifies algorithm effectiveness for solving $jsp \in X$.

The core objective is to establish a function $S : X \to A$ that, for each problem instance $jsp \in X$, selects the optimal algorithm from A based on metric y. To effectively characterize each jsp, a feature set F is constructed to represent p and assist in the decision-making process for S. Consequently, S is defined as a composite function $S = T \circ G$, where $G : X \to \mathbb{R}^{|F|}$ maps p to its feature vector in $\mathbb{R}^{|F|}$, and $T : \mathbb{R}^{|F|} \to A$ selects the algorithm from A based on this feature representation. The choice of F is critical as it must be informative and accurately represent the characteristics of JSP.

Considering the modeling of algorithm selectors in Figure 1, an algorithm selector structure is proposed, where it can be seen that it is composed of a training phase, in which the features of a set of instances are processed to generate a set of data and are solved using three solvers: GECODE, CPLEX, and GUROBI. Once the instances have been solved, the extracted features are related to the best algorithm that has solved that instance, and different machine learning models are trained in order to validate which is the one that obtains the best accuracy and thus use it to recommend future instances.

The following subsections detail each of the training processes.

3.1. Feature processing

For each instance, we extract the typical characteristics of a JSP (Job Shop Scheduling) problem, such as the number of jobs $|J|$, the number of machines $|M|$, the type of Release date, and Due date constraint Rd/Dd, and the number of speeds $|S|$. Additionally, we extract other features that are obtained in a less direct manner and aim to be as informative as possible about the complexity of the instance they represent. The extra features extracted are:

$$\max(P) \quad (16)$$

$$\mathrm{mean}(P) \quad (17)$$

$$\min(P) \quad (18)$$

The maximum processing time (16) represents the longest time required to complete any single operation within the job set. The mean processing time (Equation 17) gives the average duration of the operations, providing an overall sense of the job length. The minimum processing time (Equation 18) shows the shortest time needed for any operation, indicating the fastest job segment.

$$\max(E) \quad (19)$$

$$\mathrm{mean}(E) \quad (20)$$

$$\min(E) \quad (21)$$

The maximum energy consumption (Equation 19) indicates the highest energy required for any single operation. The mean energy consumption (Equation 20) provides the average energy used across all operations, reflecting the overall energy profile. The minimum energy consumption (Equation 21) shows the lowest energy usage for an operation, highlighting the least energy-intensive job segment.

$$\sum_{j \in J} \left(\sum_{m \in M} \max_{s \in S}(P_{jms}) \right) \quad (22)$$

Maximum makespan (Equation 22) represents the maximum makespan of the instance obtained, assuming that all operations are performed serially with their maximum processing time. This value gives the longest possible duration to complete all jobs, assuming no parallel processing.

$$\max_{j \in J} \left(\sum_{m \in M} \min_{s \in S}(P_{jms}) \right) \quad (23)$$

Minimum makespan (Equation 23) represents the makespan of the solution obtained by considering that the operations can be performed in parallel and do not overlap. This makespan represents a lower bound of the possible makespan in a solution, indicating the shortest time to complete all jobs if perfectly parallelized.

$$\sum_{m \in M} \left(\sum_{j \in J} \max_{s \in S}(E_{jms}) \right) \quad (24)$$

$$\sum_{m \in M} \left(\sum_{j \in J} \min_{s \in S}(E_{jms}) \right) \quad (25)$$

The sum of the maximum (Equation 24) and minimum (Equation 25) energy consumption is obtained by adding for each operation its maximum and minimum energy consumption, respectively. These values provide insights into the total energy requirements of the job set under extreme conditions.

$$\begin{cases} -1 & Rd/Dd = 0 \\ \sum_{j \in J} \left(\sum_{m \in M} \max_{s \in S}(P_{jms}) \right) & Rd/Dd = 1 \end{cases} \quad (26)$$

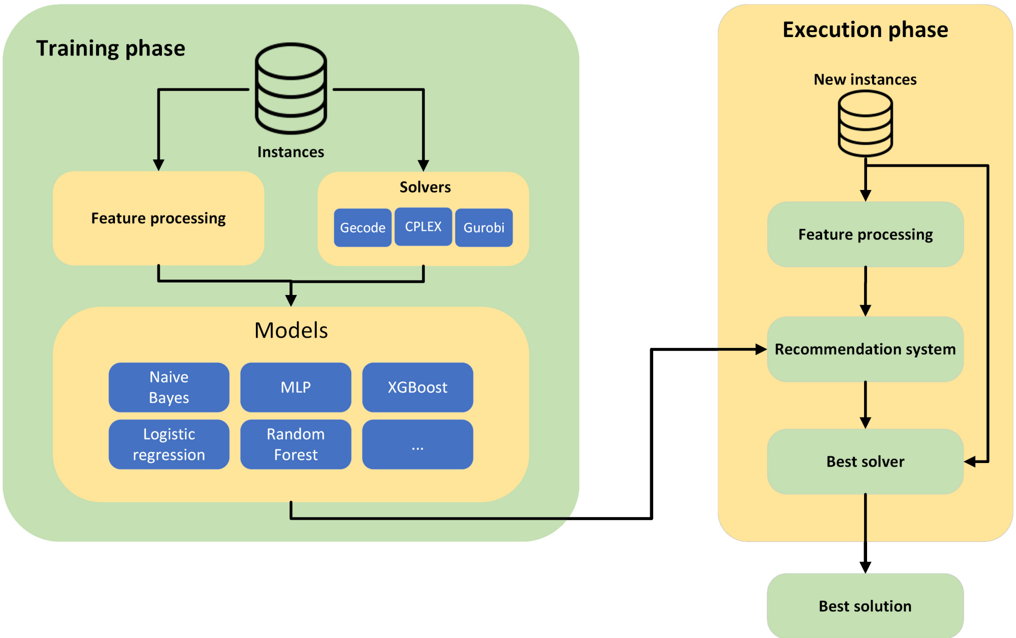


Figure 1: Structure of the proposed recommender system.

$$\begin{cases} -1 & Rd/Dd = 0 \\ \dfrac{\sum\limits_{\substack{j_1,j_2 \in J \\ j_1 \neq j_2}} \dfrac{\max(0, \min(Dd_{j_1}, Dd_{j_2}) - \max(Rd_{j_1}, Rd_{j_2}))}{Dd_{j_1} - Rd_{j_1}}}{|J| \cdot (|J|-1)} & Rd/Dd = 1 \\ \dfrac{\sum\limits_{\substack{j_1,j_2 \in J \\ j_1 \neq j_2}} \sum\limits_{m \in M} \dfrac{\max(0, \min(Dd_{j_1 m}, Dd_{j_2 m}) - \max(Rd_{j_1 m}, Rd_{j_2 m}))}{Dd_{j_1 m} - Rd_{j_1 m}}}{|J| \cdot (|J|-1) \cdot |M|} & Rd/Dd = 2 \end{cases} \tag{28}$$

Maximum Tardiness (Equation 26) represents the maximum possible delay in a solution. If there are no release or due date constraints ($Rd/Dd = 0$), it is set to -1. Otherwise, it sums the maximum processing times, indicating the worst-case delay scenario.

$$\begin{cases} -1 & Rd/Dd = 0 \\ \dfrac{\sum\limits_{j \in J} \dfrac{Dd_j - Rd_j}{\sum_{m \in M} P_{jm}}}{|J|} & Rd/Dd = 1 \\ \dfrac{\sum\limits_{j \in J} \sum\limits_{m \in M} \dfrac{Dd_{jm} - Rd_{jm}}{P_{jm}}}{|J| \cdot |M|} & Rd/Dd = 2 \end{cases} \tag{27}$$

Time-Window (Equation 27) represents the number of times a job or operation can be performed within its time window. This metric varies based on the type of release and due date constraints: no constraints, job-level constraints, or operation-level constraints, indicating flexibility in scheduling.

Overlap (Equation 28) represents the degree of overlap between the time windows of the jobs or operations. This metric assesses how much the scheduling windows for different jobs or operations coincide, which impacts the complexity and difficulty of scheduling.

3.2. Machine Learning Models

Once the instances have been vectorized and solved, a tabular data set is constructed with the characteristics of each instance and the solver that has found the best solution, that is, the one that has obtained the lowest value of the objective function.

This dataset has been separated into two subsets, a training subset with a size of 80% and a test subset with the remaining 20%. In addition, it has been ensured that the same number in proportion of instances exists in the two subsets.

The training dataset has been used to validate different models using five-fold cross-validation. The validated models are the following:

- Logistic Regression: This is a statistical method for analyzing a dataset in which one or more independent variables determine an outcome. The outcome is measured with a dichotomous variable (i.e., two possible outcomes). Logistic regression is particularly useful for binary classification problems and provides insights into the relationships between the variables and the probability of the outcomes.

- Decision Tree: This is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. A decision tree is built by splitting the dataset into subsets based on the value of input features, with the goal of making the most informative splits. This method is easy to interpret and visualize, making it useful for understanding the structure of the data.
- Gaussian Naive Bayes: This is a probabilistic classifier based on Bayes' theorem, with the assumption that the features are independent given the class label and that they follow a Gaussian distribution. Despite its simplicity, Gaussian Naive Bayes can perform well in various situations, especially when the assumption of independence roughly holds true.
- K-Nearest Neighbors (KNN): This is a non-parametric method used for classification and regression. For classification, the input consists of the k closest training examples in the feature space, and the output is a class membership. The object is assigned to the class most common among its k nearest neighbors.
- Random Forest: This is an ensemble learning method for classification and regression that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests improve the predictive accuracy and control over-fitting by averaging multiple trees, reducing the model's variance.
- XGBoost [14]: This is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the gradient boosting framework, which builds models in a stage-wise fashion and generalizes them by optimizing for a differentiable loss function. XGBoost is known for its speed and performance, making it a popular choice for structured/tabular data.
- Multi-Layer Perceptron (MLP): This is a class of feedforward artificial neural networks that consist of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node (or neuron) uses a nonlinear activation function. MLPs are capable of learning complex mappings from inputs to outputs and are trained using backpropagation.

4. Evaluation

All experiments were conducted on a system equipped with an Intel 3.60 GHz 12th generation Core i7 CPU and 64 GB of RAM. The implementation was developed in Python 3.11. Well-known solvers such as GUROBI [15], CPLEX [16], and GECODE [17], which are implemented on Minizinc, were utilized.

To evaluate the quality of the solutions obtained, the mono-objective function shown in Equation 11 is used to compare the best solutions from the solvers. Other important results, such as the average objective function, solving time, optimum, satisfaction rate, and the number of unsolved solutions, are presented above.

4.1. Instances

Instance creation is one of the most important aspects of evaluation as it allows a specific number of instances to be configured to ensure the most comprehensive evaluation possible, taking into account all possible combinations the problem may encounter in real-life scenarios.

The JSP Benchmark used for testing is composed of the number of jobs (J) and machines (M) to determine each job's tasks. These variables can take any natural number. In this test set, the set $\{5, 10, 20, 25, 50, 100\}$ is considered for J. The release and due date can take values $\{0, 1, 2\}$, speed scaling can take values $\{1, 3, 5\}$, and statistical distributions considered are $\{$*uniform*, *normal*, *exponential*$\}$. For each configuration, 10 instances are generated with different seeds to ensure substantial variation between them. Therefore, a total of $6(J) \times 6(M) \times 3(rddd) \times 3(ss) \times 3(dist) \times 10(Q) = 9720$ instances are obtained.

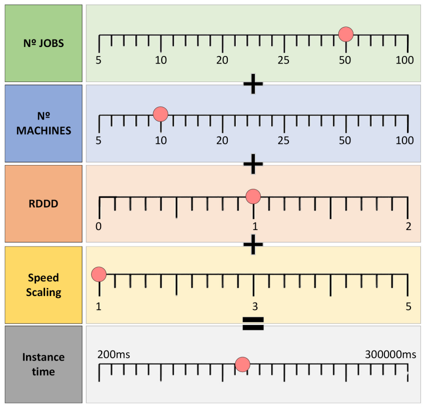


Figure 2: Distribution of timeout and relationship between time and energy.

The time allocated for resolving each instance depends on the specific characteristics of the problem. An example of this allocation is illustrated in Figure 2. In this example, an instance with 50 jobs, 10 machines, no Release Date or Due Date, and a single speed per machine is allocated 149 seconds. The principle is that if the maximum allocation time for an instance is 300,000 milliseconds, each characteristic's impact on the allocation should be equivalent. Therefore, each characteristic contributes at most $300,000/4 = 75,000$ milliseconds. In this manner, for the given example, if the Release Date and Due Date are assigned at the operation level (RDDD = 2), they contribute 75,000 milliseconds. If they are absent (RDDD = 0), they contribute 50 milliseconds. When assigned at the job level, the contribution is determined by exponential interpolation between these two cases.

4.2. Results

Upon defining the problem instances and setting appropriate search time limits for each solver, the focus shifted toward analyzing and interpreting the outcomes. This involved evaluating the efficacy of the solvers employed, assessing solution quality, and considering the broader implications within the problem domain.

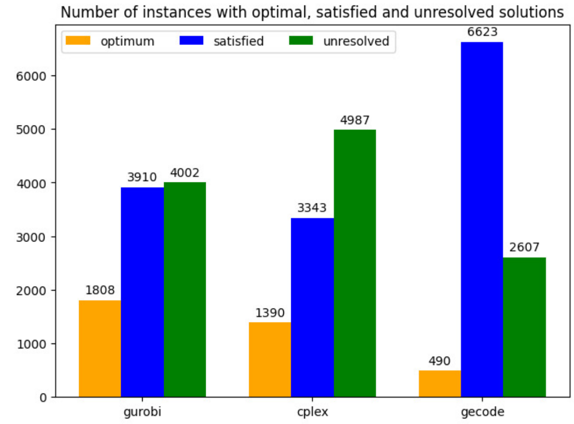


Figure 3: Quantity of optimum, satisfied and unresolved instances by each solver.

Figure 3 illustrates the distribution of solved instances among GUROBI, CPLEX, and GECODE across three categories: optimal (best solution), satisfied (feasible but not optimal), and unresolved (not solved).

GUROBI emerged as the top performer overall, solving the highest number of optimal solutions and consistently demonstrating its ability to find acceptable solutions even when optimal ones were unfeasible. This underscores GUROBI's robust capability in efficiently managing a diverse array of problem types. Conversely, GECODE excelled in finding feasible solutions, significantly outperforming other solvers in achieving satisfactory solutions. Moreover, GECODE showed the fewest instances left unresolved, highlighting its reliability in tackling complex problems without abandoning them.

In contrast, CPLEX, while proficient, faced challenges with more complex problem instances, leading to a higher incidence of unresolved cases. Although it achieved reasonable numbers of optimal and satisfactory solutions, its performance consistency was observed to be less reliable compared to GUROBI and GECODE.

Table 1 compares solution times and objective function values from GUROBI, CPLEX, and GECODE across different job and machine configurations. This analysis reveals insights into each solver's performance characteristics, highlighting strengths and limitations in solving optimization problems.

For 5 to 20 jobs, GUROBI consistently shows shorter solution times and competitive objective values compared to CPLEX and GECODE. Its efficiency and precision make it highly effective in simpler problem instances.

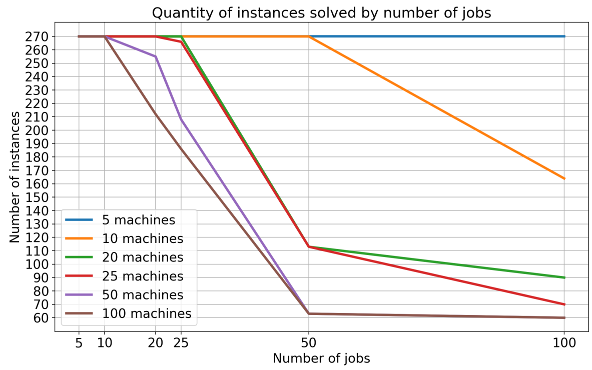
In medium-sized scenarios (20 to 50 jobs), GUROBI maintains an edge, particularly with fewer machines, though CPLEX occasionally performs better in specific configurations. GUROBI generally achieves superior objective function values in varied problem setups.

In complex cases (50 to 100 jobs), GECODE demonstrates exceptional scalability despite encountering timeouts in some instances. GUROBI and CPLEX struggle more often with timeouts as problem size and complexity increase, yet GUROBI often maintains competitive objective values.

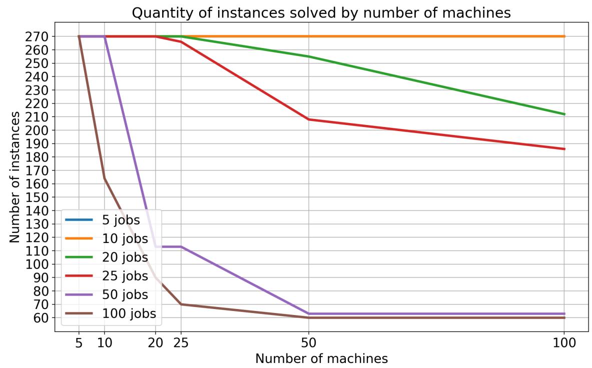
These insights underscore the importance of selecting solvers based on problem specifics. GUROBI excels in smaller to medium-sized instances, balancing efficiency and high-quality solutions. CPLEX performs well in certain medium-sized setups but faces scalability challenges. GECODE shines in complex problems, offering robust scalability and reliability despite occasional computational hurdles. These findings aid practitioners in optimizing solver choices and considering trade-offs between solution quality, efficiency, and problem complexity.

4.3. Complexity analysis

Observing the results obtained by the methods used, a relationship is observed between the parameters employed and the complexity of the instances. This part of the study focuses on the in-depth analysis of each parameter to observe its contribution to the overall complexity of the instances.



(a) by number of machines



(b) by number of jobs

Figure 4: Number of instances solved

To delve deeper into the data presented in Table 1, Figure 4 provides a general overview of the number of solved instances. Organizing the data by the number of machines, as shown in subfigure 4a, it is evident that as the number of machines increases, the number of solved instances progressively decreases except for the case of 5 machines, where all instances are solved for all possible job configurations. Looking at subfigure 4b, which is organized by the number of jobs for all possible machine configurations, it can be seen that all instances are solved for 5 and 10 jobs, but there is a notable decrease for the rest. Focusing on the set of 20 and 25 jobs, it can be observed that there is a slight decrease from 25 machines onwards; later, the reasons for this are analyzed. For instances with 50 and 100 jobs, the decrease in the number of solved instances is exponential. Only all instances are solved for a configuration of 5 machines for 100 jobs and 5 and 10 machines for 50 jobs. This figure

Jobs	Machines	Solve Time			Objective		
		GUROBI	CPLEX	Gecode	GUROBI	CPLEX	Gecode
5	5	**6499.47**	9393.54	76754.74	0.58053	**0.57924**	0.7748
	10	**22177.73**	35193.18	77122.47	0.46884	**0.46842**	0.64781
	20	**50222.29**	57898.63	81637.16	**0.42377**	0.42422	0.58867
	25	**55920.19**	59732.08	79771.76	**0.41208**	0.41279	0.5677
	50	**69226.88**	71453.81	95645.56	**0.39632**	0.40691	0.53307
	100	**110177.22**	123730.15	154180.72	**0.38944**	0.42882	0.51176
10	5	**72263.13**	76261.31	85388.49	**0.56284**	0.56433	0.77965
	10	**69831.23**	71671.93	86539.69	**0.44524**	0.4482	0.64365
	20	**71619.72**	74610.4	89860.43	**0.38908**	0.39532	0.5713
	25	**73930.71**	78202.9	91969.19	**0.3798**	0.39247	0.54207
	50	**89935.3**	95355.06	109638.51	**0.36431**	0.38019	0.49456
	100	**146282.37**	152079.82	168202.69	0.35029	**0.30281**	0.35159
20	5	**80190.01**	88650.74	89079.98	**0.65727**	0.67151	0.83772
	10	**80369.98**	89368.09	89895.89	**0.48249**	0.48393	0.71326
	20	91070.63	**83648.86**	92953.44	0.40615	**0.36374**	0.60325
	25	92555.52	**86596.51**	96516.25	0.38184	**0.3135**	0.58104
	50	109441.69	97752.14	**78306.1**	0.35987	0.24087	**0.20729**
	100	164169.07	161843.67	**112923.84**	0.29168	0.06245	**0.00733**
25	5	**73156.78**	86322.15	91381.59	0.64018	**0.63401**	0.84305
	10	**78504.99**	83658.47	92142.26	0.57263	**0.46831**	0.73123
	20	99400.75	**83989.35**	95774.25	0.46229	**0.36985**	0.62663
	25	104469.18	**76519.47**	102202.35	0.42563	**0.30161**	0.61462
	50	118524.24	127129.68	**59800.26**	0.31818	0.21552	**0.13974**
	100	168354.77	*Timeout*	**113885.23**	0.28319	*Timeout*	**0.01021**
50	5	101811.91	**86469.46**	110038.97	0.70258	**0.49251**	0.86043
	10	115365.19	**71684.29**	110542.27	0.56627	**0.21168**	0.75717
	20	108631.5	117752.67	**78930.63**	0.37003	**0.12673**	0.66168
	25	105251.06	120220	**72639.27**	0.43698	**0.16123**	0.57382
	50	136252.05	*Timeout*	**59343.32**	0.47363	*Timeout*	**0.14455**
	100	153826.43	*Timeout*	**122941.2**	0.91743	*Timeout*	**0.04168**
100	5	*Timeout*	**114685.56**	183151.94	*Timeout*	**0.32913**	0.86754
	10	*Timeout*	*Timeout*	**180057.66**	*Timeout*	*Timeout*	**0.77187**
	20	*Timeout*	*Timeout*	**139246.52**	*Timeout*	*Timeout*	**0.68049**
	25	*Timeout*	*Timeout*	**122740.76**	*Timeout*	*Timeout*	**0.59173**
	50	*Timeout*	*Timeout*	**116506.13**	*Timeout*	*Timeout*	**0.4602**
	100	*Timeout*	*Timeout*	**157021.85**	*Timeout*	*Timeout*	**0.13594**

Table 1
Comparison of mean resolution time and mean objective function obtained with different solvers.

illustrates how the number of jobs and machines affects the possibility of obtaining a solution to the problem at hand.

4.4. Algorithm selector results

Model	Accuracy (%)
Logistic Regression	76.08
Gaussian Naive Bayes	48.93
Decision Tree	79.48
K-Nearest Neighbors	78.34
Random Forest	82.87
XGBoost	**83.26**
MLP	82.81

Table 2
Table showing the training results of the different models

Table 2 shows the validation results of the tested models. As can be seen, XGBoost is the model with the best validation accuracy. Training this model with the total training data set and testing it with the test set finally yields an accuracy of 84.51%. This indicates that XGBoost performs well during the validation phase and generalizes effectively to unseen data. The high accuracy suggests that XGBoost's ensemble learning approach, which combines multiple decision trees to improve performance, is particularly well-suited to this dataset. Moreover, the performance difference between XGBoost and other models like Random Forest and MLP, which also showed strong results,

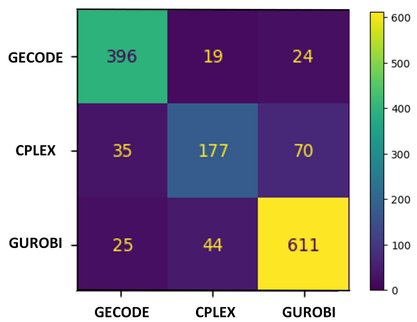


Figure 5: Confusion matrix for the algorithm selector predictions (true algorithms on the y-axis, predicted algorithms on the x-axis).

The confusion matrix in Figure 5 presents the algorithm selector's classification results. Each cell value represents the number of instances where the algorithm selector predicted the algorithm in the corresponding column for a problem best solved by the algorithm in the corresponding row. For instance, the selector correctly identified GUROBI for 611 out of the total instances where GUROBI was the best choice. Similarly, GECODE was correctly identified 396 times. However, there are misclassifications, such as predicting GUROBI when CPLEX was optimal, which occurred 70 times.

An in-depth analysis of the precision and recall metrics provides further insights into the performance of the algorithm selector. GECODE achieved a precision of 90.20%, indicating that 90.20% of the instances predicted as GECODE were correctly identified. Its recall was 86.84%, signifying that 86.84% of the actual GECODE instances were correctly detected. This high precision and recall demonstrate the algorithm selector's robustness in identifying GECODE instances accurately.

On the other hand, CPLEX showed a precision of 62.76%, meaning that only 62.76% of the predictions for CPLEX were accurate, and a recall of 73.75%, which indicates that 73.75% of the actual CPLEX instances were correctly classified. The lower precision for CPLEX suggests a higher rate of false positives, which could imply that the algorithm selector often misclassifies other algorithms as CPLEX.

For GUROBI, the precision was 89.85%, reflecting that 89.85% of the GUROBI predictions were correct, and the recall was 86.66%, meaning that 86.66% of the actual GUROBI instances were identified correctly. These values indicate a strong performance, similar to GECODE, highlighting the selector's efficiency in recognizing GUROBI accurately.

These metrics, precision, and recall, are crucial for evaluating the algorithm selector's effectiveness, as they provide a more comprehensive understanding of its performance beyond simple accuracy. They highlight the selector's strengths in accurately identifying certain algorithms while also pointing out areas where misclassification occurs, thus providing a clear direction for further improvements.

5. Conclusions

This study explores the complexities of JSP, emphasizing its NP-completeness and diverse optimization goals such as makespan, energy consumption, and tardiness. The problem presents significant computational challenges due to its combinatorial nature, making timely optimal solutions crucial in operations research and manufacturing.

An innovative aspect of this research is integrating machine learning techniques to enhance algorithm selection for JSP instances. By extracting comprehensive features like job and machine characteristics, release dates, and energy requirements, models such as XGBoost and Random Forest were effectively trained. These models accurately recommend suitable solvers like GUROBI, CPLEX, and GECODE, streamlining decision-making for solving diverse and complex scheduling problems.

GUROBI proved particularly efficient for smaller to medium-sized instances, consistently delivering optimal and satisfactory solutions across different configurations. Meanwhile, GECODE demonstrated robust scalability, excelling in complex scenarios despite occasional computational challenges. This analysis underscores the importance of selecting solvers based on specific problem parameters to optimize solution quality and computational efficiency.

Looking ahead, the study suggests refining feature extraction methodologies to enhance the algorithm selector's accuracy across a broader range of JSP scenarios. Advancements in solver performance under varying constraints promise to expand the practical utility of scheduling optimization tools in real-world manufacturing settings, emphasizing efficiency and sustainability.

Although the results obtained are not as high as those reported in the literature, it should be noted that the energy-aware JSP constitutes a more complex problem compared to the standard JSP and flexible JSP found in the literature. Furthermore, this study uses a smaller set of features than those used in other studies, yet the accuracy achieved is not significantly lower.

In conclusion, this work advances both academic understanding and practical applications by integrating traditional optimization techniques with modern machine-learning approaches. It offers tools that can significantly benefit research and industrial practices, addressing contemporary challenges in operations management and manufacturing logistics.

References

[1] H. Xiong, S. Shi, D. Ren, J. Hu, A survey of job shop scheduling problem: The types and models, Computers & Operations Research 142 (2022) 105731. doi:https://doi.org/10.1016/j.cor.2022.105731.

[2] D. B. M. M. Fontes, S. M. Homayouni, J. F. Gonçalves, A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources, European Journal of Operational Research 306 (2023) 1140–1157. doi:https://doi.org/10.1016/j.ejor.2022.09.006.

[3] D. M. Torres, F. Barber, M. A. Salido, Psplib-energy: a extension of psplib library to assess the energy optimization in the rcpsp, Inteligencia Artificial 17 (2014) 48–61. doi:10.4114/intartif.vol17iss54pp48-61.

[4] M. M. S. El-Kholany, M. Gebser, K. Schekotihin, Problem decomposition and multi-shot asp solving for job-shop scheduling, 2022.

[5] C. Perez, M. A. Salido, D. Gurrea, A metaheuristic search technique for solving the warehouse stock management problem and the routing problem in a real company, in: Artificial Intelligence XXXVII, Springer International Publishing, Cham, 2020, pp. 187–201.

[6] M. Dai, D. Tang, A. Giret, M. A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, Robotics and Computer-Integrated Manufacturing 59 (2019) 143–157. doi:https://doi.org/10.1016/j.rcim.2019.04.006.

[7] C. Pérez, L. Climent, G. Nicoló, A. Arbelaez, M. A. Salido, A hybrid metaheuristic with learning for a real supply chain scheduling problem, Engineering Applications of Artificial Intelligence 126 (2023) 107188. doi:https://doi.org/10.1016/j.engappai.2023.107188.

[8] M. A. Salido, J. Escamilla, A. Giret, F. Barber, A genetic algorithm for energy-efficiency in job-shop scheduling, Int J Adv Manuf Technol 85 (2016) 1303–1314. doi:https://doi.org/10.1007/s00170-015-7987-0.

[9] K. Jyothi, R. B. Dubey, Minimizing non-processing en-

ergy consumption/total weighted tardiness & earliness, and makespan into typical production scheduling model-the job shop scheduling problem, Journal of Intelligent & Fuzzy Systems 45 (2023) 6959–6981. doi:10.3233/JIFS-222362.

[10] A. Ham, M.-J. Park, K. M. Kim, Energy-aware flexible job shop scheduling using mixed integer programming and constraint programming, Mathematical Problems in Engineering 2021 (2021) 1–12. doi:10.1155/2021/8035806.

[11] D. Müller, M. G. Müller, D. Kress, E. Pesch, An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning, European Journal of Operational Research 302 (2022) 874–891. doi:https://doi.org/10.1016/j.ejor.2022.01.034.

[12] S. Strassl, N. Musliu, Instance space analysis and algorithm selection for the job shop scheduling problem, Computers & Operations Research 141 (2022) 105661. doi:https://doi.org/10.1016/j.cor.2021.105661.

[13] J. R. Rice, The algorithm selection problem. this work was partially supported by the national science foundation through grant gp-32940x. this chapter was presented as the george e. forsythe memorial lecture at the computer science conference, february 19, 1975, washington, d. c., in: Advances in Computers, volume 15 of *Advances in Computers*, Elsevier, 1976, pp. 65–118. doi:https://doi.org/10.1016/S0065-2458(08)60520-3.

[14] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, 2022, p. 785–794. doi:10.1145/2939672.2939785.

[15] Y. Zeiträg, J. Rui Figueira, G. Figueira, A cooperative coevolutionary hyper-heuristic approach to solve lot-sizing and job shop scheduling problems using genetic programming, International Journal of Production Research (2024). doi:10.1080/00207543.2023.2301044.

[16] D. Rooyani, F. M. Defersha, An efficient two-stage genetic algorithm for flexible job-shop scheduling, IFAC-PapersOnLine 52 (2019). doi:10.1016/j.ifacol.2019.11.585.

[17] L. Ingmar, C. Schulte, Making compact-table compact, in: Principles and Practice of Constraint Programming, Springer International Publishing, Cham, 2018, pp. 210–218.

Instance Configuration for Sustainable Job Shop Scheduling

Christian Pérez[1,2,*,†], Carlos March[1,†] and Miguel Salido[1]

[1]Universitat Politècnica de València, Instituto de Automática e Informática Industrial, Camì de Vera S/N, Valencia, Spain

[2]Universitat Politècnica de València, valgrAI - Valencian Graduate School and Research Network of Artificial Intelligence, Camì de Vera S/N, Valencia, Spain

Abstract

The Job Shop Scheduling Problem (JSP) is a pivotal challenge in operations research and is essential for evaluating the effectiveness and performance of scheduling algorithms. Scheduling problems are a crucial domain in combinatorial optimization, where resources (machines) are allocated to job tasks to minimize the completion time (makespan) alongside other objectives like energy consumption. This research delves into the intricacies of JSP, focusing on optimizing performance metrics and minimizing energy consumption while considering various constraints such as deadlines and release dates. Recognizing the multi-dimensional nature of benchmarking in JSP, this study underscores the significance of reference libraries and datasets like JSPLIB in enriching algorithm evaluation. The research highlights the importance of problem instance characteristics, including job and machine numbers, processing times, and machine availability, emphasizing the complexities introduced by energy consumption considerations.

An innovative instance configurator is proposed, equipped with parameters such as the number of jobs, machines, tasks, and speeds, alongside distributions for processing times and energy consumption. The generated instances encompass various configurations, reflecting real-world scenarios and operational constraints. These instances facilitate comprehensive benchmarking and evaluation of scheduling algorithms, particularly in contexts of energy efficiency. A comprehensive set of 500 test instances has been generated and made publicly available, promoting further research and benchmarking in JSP. These instances enable robust analyses and foster collaboration in developing advanced, energy-efficient scheduling solutions by providing diverse scenarios.

Keywords

Job Shop Scheduling Problem, Instance Generation, Benchmarking, Energy consumption, Speed scaling

1. Introduction

The Job Shop Scheduling Problem (JSP) stands as a cornerstone in the realm of operations research and optimization, representing a fundamental challenge pivotal for evaluating algorithmic effectiveness and performance. In essence, JSP revolves around the intricate task of allocating jobs to machines within a manufacturing environment to optimize a plethora of performance metrics, ranging from makespan and flow time to tardiness, resource utilization, and energy consumption [1]. The process of benchmarking in JSP is multi-dimensional, necessitating the definition and evaluation of metrics such as makespan, energy consumption, and tardiness to assess scheduling efficiency and resource utilization [2]. Reference libraries and datasets like JSPLIB play an indispensable role in these benchmarking endeavours, furnishing researchers with a rich array of instances sourced from seminal works and experimental studies, thereby enriching the evaluation of algorithms [3].

A profound understanding of problem instance characteristics significantly shapes benchmarking efforts in JSP. Factors such as the number of jobs and machines, variability in processing times, machine availability, and precedence relationships exert notable influences on algorithm performance [4]. Furthermore, incorporating energy consumption considerations, contingent upon machine speed and operational attributes, introduces an added layer of complexity to these instances [5]. The delicate balance between energy consumption and scheduling decisions emerges as paramount in achieving energy efficiency goals without compromising production targets [6].

In recent years, the spotlight on addressing energy efficiency within the realm of JSP has intensified, driven by its profound environmental and economic implications [7]. Strategies involving integrating speed-adjustable machines and vehicles have been explored as avenues to optimize energy consumption while upholding productivity levels [8]. Concurrently, developing advanced algorithms and optimization techniques tailored to tackle energy-related challenges has seen significant advancements, considering factors such as machine speed, idle time, and energy requirements [9]. Real-world implementations of these energy-efficient strategies have yielded tangible benefits, manifesting in substantial cost savings

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.
† These authors contributed equally.
cripeber@upv.es (C. Pérez); cmarmoy@upv.es (C. March); msalido@upv.es (M. Salido)
https://gps.blogs.upv.es/ (C. Pérez); https://gps.blogs.upv.es/ (C. March); https://gps.blogs.upv.es/ (M. Salido)
0000-0002-9121-7939 (C. Pérez); 0009-0009-7525-9133 (C. March); 0000-0002-4835-4057 (M. Salido)

and positive environmental impacts [10].

In conclusion, the imperative of energy efficiency in JSP research has become increasingly pronounced, paralleling the traditional focus on performance metrics [11]. Benchmarking is a linchpin in evaluating the efficacy of energy-efficient scheduling strategies, furnishing invaluable insights into their ramifications on production efficiency and energy consumption [12]. Manufacturers can embark toward more sustainable and economically viable operations by harnessing advanced optimization techniques and leveraging real-world implementations.

2. Instance Configuration

Test sets play a vital role in JSP research by providing a standardized platform for comparing algorithmic approaches. Their diversity enables researchers to evaluate various algorithms, from heuristics to exact methods, identifying strengths, weaknesses, and potential limitations across different contexts [13]. However, as industrial systems evolve, the complexity of real-world problems increases, necessitating the development or expansion of test sets to simulate real-world scenarios better.

To address this need, the proposed instance configurator operates with the following parameters:

- $J = \{0, \ldots, n\}$: the set of jobs.
- $M = \{0, \ldots, m\}$: the set of machines.
- $S = \{0, \ldots, s\}$: the set of speeds, indexed by s in S.
- T_j: the set of tasks in job j, indexed by $t_{jt} \in T_j,\ \forall j \in J,\ \forall t \in M$.
- D_{jt}: the due date of task job t_{jt} $\forall j \in J,\ \forall t \in M$.
- R_{jt}: the release date of task job t_{jt} $\forall j \in J,\ \forall t \in M$.
- P_{jts}: the processing time of task job $t_{jt},\ \forall j \in J,\ \forall t \in M,\ \forall s \in S$.
- E_{jts}: the energy consumption for processing task job $t_{jt},\ \forall j \in J,\ \forall t \in M,\ \forall s \in S$.

The process of configuring instances, managed by Algorithm 1, is initiated by receiving several input parameters: the number of instances to generate Q, the number of machines M, the count of jobs J, the number of tasks t, the types of release and due dates $rrdd$, random seeds $seeds$, and the distribution $dist$.

Subsequently, the algorithm executes a series of steps to generate instances systematically. The random seed is initialized to ensure reproducibility, and an empty list G is created to store the generated instances (Line 2). The

Algorithm 1 Instance Configurator

input: Quantity of instances Q, Number of machines M, Number of jobs J, Number of tasks T, Release and due date type $rrdd$, Random seed $seed$, Distribution $dist$
output: Generated instances G

1: $SetSeed(seed)$
2: $G \leftarrow [\ ]$
3: **for** q from 1 to Q **do**
4: $\quad O \leftarrow GenerateJobsOperations(T, J, M)$
5: $\quad P \leftarrow GenerateProcessingTimes(J, M, S)$
6: $\quad E \leftarrow GenerateEnergy(J, M, S)$
7: $\quad R, D \leftarrow GenerateRDDate(J, M, dist)$
8: $\quad G \leftarrow G \cup JSP(O, P, E, R, D)$
9: **end for**
10: **return** G

algorithm generates jobs and tasks within a loop iterating through each instance q from 0 to $Q-1$ (Line 4).

Next, each job operation's processing times and energy consumption are generated.

$$f(x) = \lfloor e^{-\frac{x}{100}} \times 100 \rfloor \tag{1}$$

$$g(x) = 4.0704 \times \frac{\log(2)}{\log(1 + (x \times 2.5093)^3)} \tag{2}$$

The functions responsible for generating the processing times (line 5) and energy consumption (line 6) for the combination of jobs, machines, and speeds are managed by functions $f(x)$ 1 and $g(x)$ 2, as illustrated in Figure 1 as studied in [14]. These functions balance the variables to establish a correlation between processing time and energy consumption. In these equations, x represents the processing time for Equation $f(x)$ 1 and the energy consumption for Equation $g(x)$ 2. Speeds are generated by obtaining the energy consumption percentage for each speed using Equation $f(x)$ 1, which models an inverse relationship between processing time and energy consumption. This involves dividing the interval [0.5, 3] into $|S| - 1$ equal parts, where the boundaries of these new intervals correspond to the energy consumption percentages for each speed. Subsequently, Equation $g(x)$ 2 is utilized to determine the fraction of time corresponding to each speed.

Release and due dates are computed using functions based on the chosen distribution (Line 7). Each distribution offers distinct characteristics suited for modeling various real-world scenarios.

The exponential distribution, defined by its probability density function $f(x; \lambda) = \lambda e^{-\lambda x}$, is ideal for modeling the time until an event occurs, such as machine failures or job arrivals, assuming a constant hazard rate $\lambda > 0$. Its mean is $\frac{1}{\lambda}$. The Gaussian distribution, with mean μ and standard deviation σ, has a probability density function

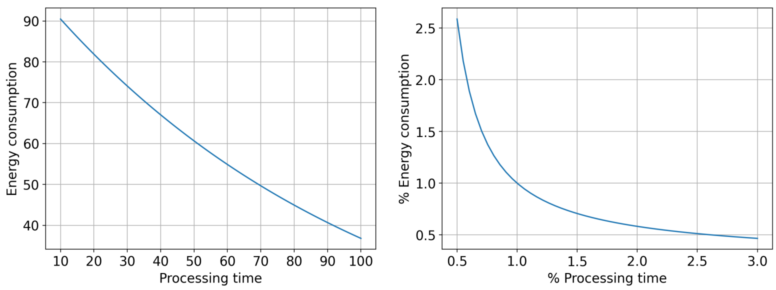


Figure 1: Distribution of processing times and the relationship between time and energy

$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. This distribution represents naturally occurring variations in processing times or delays. The uniform distribution generates values evenly within a specified range, defined by the probability density function $f(x;a,b) = \frac{1}{b-a}$, where a and b are the lower and upper bounds, respectively. It provides a straightforward way to explore a range of scenarios without bias towards any particular value. These distributions were selected due to their unique properties and common use in modeling different real-world data types, enhancing the diversity and comprehensiveness of the generated instances [15].

Additionally, a random start is chosen for each job within a specified range for the release and due date intervals. The time interval between release and due dates is determined based on the median processing time. A random value within the corresponding interval is generated depending on the chosen distribution. This comprehensive approach ensures the creation of instances encompassing a wide range of scenarios, facilitating robust analyses and evaluations.

These steps culminate in constructing a JSP instance (Line 8), incorporating the generated data. Finally, the algorithm returns the list G containing the generated instances. This systematic approach ensures the creation of instances that cover diverse scenarios, which is crucial for comprehensive analyses and evaluations.

3. Generated Problems

A comprehensive set of random instances has been generated following the procedure described in Algorithm 1. These instances exhibit diverse characteristics: the number of jobs ranges from thirty to two hundred fifty, and the number of machines ranges from three to twenty. Normal, exponential, and uniform distributions were utilized in the generation process.

Each instance was extended by relaxing release and due date restrictions. Leveraging three different speed scales, variations of each problem were created, maintaining identical data but with different operational speeds. Specifically, two additional instances were derived from each original: one incorporating the first, third, and fifth-speed scaling and another utilizing only the third-speed scaling.

In addition to varying job and machine counts, the instances encompass different operational complexities and constraints. For instance, some involve jobs with precedence constraints, necessitating certain jobs to be finished before others begin. This complexity challenges algorithms to find optimal solutions efficiently. The chosen distributions—normal, exponential, and uniform—offer a spectrum of scenarios, ranging from predictable and evenly spread job times to highly variable and unpredictable duration. This diversity ensures that the generated instances serve as robust benchmarks to evaluate the performance of scheduling algorithms under varied conditions.

Moreover, a collection of 500 test instances has been generated and made publicly accessible through [16]. These instances incorporate mixed distributions and speed scalings, providing researchers with a comprehensive dataset to evaluate the efficacy of scheduling algorithms. The research group aims to foster collaboration and innovation in planning and scheduling research by facilitating access to these instances. Researchers can use these standardized problems to compare methods and contribute to advancing scheduling solutions.

In summary, the generated instances cover a wide range of job and machine configurations, distribution types, and speed variations, making them suitable for diverse scheduling and planning research applications. Their availability for public use enhances their utility, promoting collaboration and enabling continuous improvement and benchmarking in the field.

4. Acknowledgments

The authors gratefully acknowledge the financial support of the European Social Fund (Investing In Your Future), the Spanish Ministry of Science (project PID2021-125919NB-I00), and valgrAI - Valencian Graduate School and Research Network of Artificial Intelligence and the Generalitat Valenciana, and co-funded by the European Union.

References

[1] H. Xiong, S. Shi, D. Ren, J. Hu, A survey of job shop scheduling problem: The types and models, Computers & Operations Research 142 (2022) 105731. doi:https://doi.org/10.1016/j.cor.2022.105731.

[2] D. B. M. M. Fontes, S. M. Homayouni, J. F. Gonçalves, A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources, European Journal of Operational Research 306 (2023) 1140–1157. doi:https://doi.org/10.1016/j.ejor.2022.09.006.

[3] D. M. Torres, F. Barber, M. A. Salido, Psplib-energy: a extension of psplib library to assess the energy optimization in the rcpsp, Inteligencia Artificial 17 (2014) 48–61. doi:10.4114/intartif.vol17iss54pp48-61.

[4] M. M. S. El-Kholany, M. Gebser, K. Schekotihin, Problem decomposition and multi-shot asp solving for job-shop scheduling, 2022.

[5] C. Perez, M. A. Salido, D. Gurrea, A metaheuristic search technique for solving the warehouse stock management problem and the routing problem in a real company, in: Artificial Intelligence XXXVII, Springer International Publishing, Cham, 2020, pp. 187–201.

[6] M. Dai, D. Tang, A. Giret, M. A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, Robotics and Computer-Integrated Manufacturing 59 (2019) 143–157. doi:https://doi.org/10.1016/j.rcim.2019.04.006.

[7] C. Pérez, L. Climent, G. Nicoló, A. Arbelaez, M. A. Salido, A hybrid metaheuristic with learning for a real supply chain scheduling problem, Engineering Applications of Artificial Intelligence 126 (2023) 107188. doi:https://doi.org/10.1016/j.engappai.2023.107188.

[8] M. A. Salido, J. Escamilla, A. Giret, F. Barber, A genetic algorithm for energy-efficiency in job-shop scheduling, Int J Adv Manuf Technol 85 (2016) 1303–1314. doi:https://doi.org/10.1007/s00170-015-7987-0.

[9] K. Jyothi, R. B. Dubey, Minimizing non-processing energy consumption/total weighted tardiness & earliness, and makespan into typical production scheduling model-the job shop scheduling problem, Journal of Intelligent & Fuzzy Systems 45 (2023) 6959–6981. doi:10.3233/JIFS-222362.

[10] A. Ham, M.-J. Park, K. M. Kim, Energy-aware flexible job shop scheduling using mixed integer programming and constraint programming, Mathematical Problems in Engineering 2021 (2021) 1–12. doi:10.1155/2021/8035806.

[11] J. Kotary, F. Fioretto, P. V. Hentenryck, Fast approximations for job shop scheduling: A lagrangian dual deep learning method, AAAI 36 (2022) 7239–7246. doi:10.1609/aaai.v36i7.20685.

[12] L. Toma, M. Zajac, U. Störl, Solving distributed flexible job shop scheduling problems in the wool textile industry with quantum annealing, 2024. arXiv:2403.06699.

[13] J. Hoorn, The current state of bounds on benchmark instances of the job-shop scheduling problem, Journal of Scheduling 21 (2017) 127–128. doi:10.1007/s10951-017-0547-8.

[14] D. Morillo Torres, F. Barber Sanchís, M. A. Salido Gregorio, Psplib-energy: Una extension de la libreria psplib para la evaluacion de la optimizacion energetica en el rcpsp, Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial 17 (2014) 35–48.

[15] L. Gui, X. Li, Q. Zhang, L. Gao, Domain knowledge used in meta-heuristic algorithms for the job-shop scheduling problem: Review and analysis, Tsinghua Science and Technology null (2024) null. doi:10.26599/tst.2023.9010140.

[16] Planning, S. R. Group, Energy Instances for Sustainable Job Shop Scheduling, 2024. doi:10.5281/zenodo.11640130.

Product visualization in configurators: laying the foundations for a comparative description

Andrea Petterle[1,†], Enrico Sandrin[1,*,†] and Cipriano Forza[1,†]

[1] University of Padova, Stradella San Nicola 3, 36100 Vicenza, Italy

Abstract

Visualization plays a critical role in the purchasing process, particularly for customized products, as it improves customer engagement and decision-making. Despite the importance of product configurators in presenting customized products, there is limited research on the characteristics of visualization modes that configurators employ. This study aims to address this gap by developing an evaluation framework consisting of 11 descriptive variables: embodiment, presence, interactivity, authenticity, realism, media richness, avatar similarity, functional control, visual control, interaction richness, and vividness. Each variable of the framework is defined and exemplified by practical examples. These variables, derived from the literature on e-commerce and customer experience, offer a structured framework to describe and compare product visualization modes in configurators.

Keywords

configurator, product visualization, virtual reality, augmented reality

1. Introduction

Visualization is a crucial element in the purchasing process, especially for customized products. The configuration process demands effort from the customer but ultimately strengthens their bond with the product. According to Di et al. [1: 550], images enhance attention, trust, and conversion rates in the purchasing process. More images provide a more complete visual representation of the product, which is effective in boosting sales. Interactive and visually appealing configurators help customers make informed decisions aligned with their expectations. Thus, visualization may reduce choice complexity and improve consumer benefits from customization.

Effective visualization is a powerful tool for knowledge transfer and assimilation, having been used since ancient times [2, 3, 4]. The transfer and assimilation of visual knowledge is simpler and faster than that of textual knowledge [3, 4]. The textual representation of a system of relationships through, for example, a table may also be extremely accurate, but the absence of visualization makes the passage of information hermetic and tedious. This is an example of why highly valuable work without visualization can be difficult to understand, even for experts in the same field, not to mention professionals from other scientific areas [3]. In the case of customized products, the variant may be novel and difficult to understand to the customer, thus the use of visualization may reduce the cognitive complexity borne by the customer at the stage of selecting the most suitable configuration to satisfy his/her needs.

Visualization technologies are undergoing significant advancements. For example, Apple's Vision Pro revolutionizes visual representation by projecting high-resolution output directly into the eyes, controlled by eye tracking and gestures, eliminating traditional peripherals such as mouse and keyboards [5]. Advanced visualization methods such as augmented reality (AR), virtual reality (VR), and mixed reality (MR) are becoming more and more widespread. The AR, VR, and MR market was valued at approximately $26 billion in 2021 and is expected to reach $242 billion by 2028 [6]. Consumer trends indicate that 71% would shop more frequently using AR, 61% prefer AR-equipped stores,

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
* Corresponding author.
† These authors contributed equally.
andrea.petterle@studenti.unipd.it (A. Petterle);
enrico.sandrin@unipd.it (E. Sandrin);
cipriano.forza@unipd.it (C. Forza)
0009-0000-0041-0112 (A. Petterle);
0000-0001-9170-0683 (E. Sandrin);
0000-0003-4583-2962 (C. Forza)

and 55% find that AR makes shopping more enjoyable [7].

An important tool for presenting and selling customized products is the configurator. However, research on the visualization of configurable products is limited and a comprehensive and comparative description of the various product visualization modes in configurators is lacking [8].

This research starts to fill this gap by proposing an evaluation framework composed of 11 descriptive variables. These 11 variables, derived from customer experience and e-commerce literature, enable a systematic description of the product visualization modes that a configurator can adopt. The framework defines the 11 descriptive variables according to the literature and presents examples of their modality. The proposed framework is designed to evaluate all visualization modes used in product configurators, encompassing both traditional (e.g., 2D and 3D images) and new modes (e.g., VR, AR, and MR), serving as a consistent tool for describing and comparing different visualization modes in configurators.

2. Theoretical background

There is a need for comparative descriptions of product visualization modes in configurators. To be able to describe and compare in a structured way these product visualization modes, a framework that includes a set of description variables that allow to describe in a structured way the visualization modes is needed. With the goal of finding the dimensions of the framework along which to describe and evaluate visualization modes in configurators, the relevant literature was searched, although it did not refer to product configurators. Three sources [9, 10, 11] and their references are taken as a starting point for our research.

Zeng and Richardson [9] conducted a review of the literature on product presentation characteristics in e-commerce. This literature review synthesizes existing research findings, explores key theoretical foundations, and highlights the predominant research theories and methodologies employed in the field. It also categorizes the constructs used to describe the characteristics of presentation formats, consumer reactions and performance, as well as marketing effects, such as attitudes towards the product and purchase intention.

Flavián et al. [10] discussed the impact of VR, AR, and MR on the integration of physical and virtual objects, leading to new hybrid customer experiences. They highlighted the lack of clear boundaries between these technologies and experiences, proposing a new taxonomy, the 'EPI Cube,' to classify current and potential technologies that enhance customer experiences. The 'EPI Cube' is based on three dimensions of description: (technological) embodiment, (psychological) presence, and (behavioral) interactivity.

Recently, Hsu et al. [11] empirically investigated how key variables (interactivity, authenticity, vividness, product presence, and instant gratification) affect impulsive purchasing intentions using AR.

3. Framework for visualization description

The framework variables are taken from the work of Zeng and Richardson [9], Flavián et al. [10], and Hsu et al. [11] and the references cited in these papers. Furthermore, focused searches of literature on specific variables and real applications were performed to clearly define or to exemplify the variables considered in the present framework. Through this process, 11 key variables have been identified:

1. Embodiment
2. Presence
3. Interactivity
4. Authenticity
5. Realism
6. Media richness
7. Avatar similarity
8. Functional control
9. Visual control
10. Interaction richness
11. Vividness

Each variable in the framework is defined and exemplified by practical examples. By integrating and extending previous works, this framework provides a more comprehensive set of variables, scientifically justified by its potential to offer a more complete and nuanced understanding of diverse aspects of product visualization modes for evaluating configurators.

3.1. Embodiment

3.1.1. Definitions

"Ihde [12] regarded embodiment as situations in which technological devices mediate the user's experience and, as a consequence, the technology becomes an extension of the human body and helps to interpret, perceive and interact with one's immediate surroundings" [10: 550].

Technological embodiment involves two important factors: sensory stimulation [10, 13, 14], i.e., the process of activating and responding to the body's senses through received stimuli [13, 14]; and immersion [10, 13, 15], i.e., the ability of the technology to allow users to better focus on what is in front of them and to extend their perception of time; it can have positive effects on experience satisfaction [10, 16].

3.1.2. Examples

Intraocular lenses, which belong to the category of devices implanted in the human body, guarantee a maximum level of embodiment. They are small contact lenses that are permanently implanted in the eye in order to restore the visual capacity that has been impaired as a result of operations or pathologies affecting this apparatus [17]. Embodiment is maximized because the user is unaware that he or she is wearing a device, but benefits from countless advantages in daily life; in this case, the technology is in complete symbiosis with the human body and acts as an extension of it.

Figure 1: Intraocular lens. Photograph: Frank C. Müller [Public domain], via Wikimedia Commons. (https://commons.wikimedia.org/wiki/File:Hinterkammerlinse_01_(fcm).jpg).

3.2. Presence

3.2.1. Definitions

Vonkeman et al. [18: 1039] define presence as "the degree to which an online product experience appears to be unmediated, rather than mediated". They build upon the definition of Waterworth et al. [19], which describe "presence as a concept that provides the user with the illusion of nonmediation, is driven by external technological sensory cues, and is subjective in nature by focusing on the experience of the user" [18: 1039].

Flavián et al. [10: 551] regard presence "as a psychological stage (not related to a specific technology) and the medium is simply the way to arrive at that stage [20]". They define presence as "the user's sensation of being transported to a distinct environment outside the real human body [13]" [10: 551].

3.2.2. Examples

Presence is a highly subjective sensation and can therefore be evoked in different ways depending on the user in question. This perception can be evoked by reading a book, listening to a song, and interacting with video games [10, 21]. Books and video games allow one to identify with the story: in a detective story, one can identify with the investigator, while in an adventure game, one plays the role of the protagonist directly. Music allows us to abstract from the real world through lyrics or specific sounds, such as the rhythm in South American music, which can make us imagine being in those faraway places.

Presence can be provoked in a virtual environment when there is a sense of 'illusion of place' [10, 22]. Virtual reality systems, such as visors, can facilitate this factor by providing deep sensory stimulation that is so immersive that they can cause a nausea-like disturbance related to perceived movement within the device [23].

3.3. Interactivity

3.3.1. Definitions

Interactivity is defined as "the extent to which users can participate in modifying the form and content of a mediated environment in real time" [24: 84]. It regards "the users' capacity to modify and receive feedback to their actions in the reality where the experience is taking place [25, 26]. […] Interactivity is a behavioral factor in that users have the ability to control and manipulate the environment that is in front of them [27]" [10: 552].

The mechanistic or structuralist approach [28] "considers interactivity as the response to the attributes of the technology and proposes that it can be enhanced through the development of these technologies" [10: 552].

Interactivity is not a yes/no property: it is a matter of degree. "Degree to which users of a medium can manipulate the form or content of the mediated environment" [9: 5] [e.g., 18, 29]. In the case of AR, Hsu et al. [11: 5] define interactivity as the "degree to which users can choose, browse, and look up product information and provide feedback through an AR app [24, 30]".

3.3.2. Examples

The characteristic of interactivity is easily found in the everyday life of each individual; just think of the use of a computer or a smartphone, where a multitude of interactions take place, consisting of making changes to the system and receiving feedback on them.
In virtual reality devices, it turns out that the control of navigation is the basic level of interactivity present; think of the first prototype 'Sensorama,' which played short films and multimedia content with the involvement of multiple senses [31].

An internal tool such as the viewer can present both levels of interactivity: while watching a film, the device allows viewing in a controlled navigation mode; on the other hand, while playing a video game, using the joystick and tactile devices (such as suits and gloves), one can modify and monitor the state of the objects presented [10, 22]. The same is true for an external tool such as a computer. When buying a product on Amazon, the process is controlled and the consumer simply follows the predetermined steps. In contrast, when using a product configurator such as Nike's, the customer is able to manipulate and configure the product as he or she wishes.

3.4. Authenticity

3.4.1. Definitions

Authenticity is the “degree to which users understand a product based on their prior consumption experience, knowledge, or time and space in an AR app environment” [11: 5]. The concept of authenticity by Grayson and Martinec [32] is examined in relation to consumers' evaluation of market offerings. This approach distinguishes the two types of authenticity:

- Indexical authenticity gives the term 'authentic' the meaning of describing “something that is thought not to be a copy or an imitation [33: 400, 34: 157]” [32: 297].
- Iconic authenticity gives the term 'authentic' the meaning of describing “something whose physical manifestation resembles something that is indexically authentic. Authors sometimes distinguish this sense of authenticity from indexical authenticity by using phrases such as ‘authentic reproduction’ or ‘authentic recreation’ [33: 399, 35: 421-422, 36: 208]” [32: 297].

Algharabat and Dennis [37: 6] propose the following definition of “perceived authenticity in a computer-mediated environment: Authenticity is a psychological state in which virtual objects presented in 3D in a computer-mediated environment are perceived as actual objects in a sensory way” [37: 6].

3.4.2. Examples

According to the definition of indexical authenticity, Jimmy Stewart's handprints in the concrete of Grauman's Chinese Theatre in Los Angeles are considered authentic if they are perceived to be the actor's original and genuine handprints [32, 38]. According to the description of iconic authenticity, silver pieces available in a museum gift shop are considered authentic because they are believed to bear remarkable similarities to coins produced by Spanish colonies in the 16th century [32, 39]. Furthermore, to judge the authenticity of a Victorian-era chair, a consumer must have some basic or more detailed idea of how it looks and how it is used [32].

In general, a product image that can be considered authentic is a photograph of it, as it allows the user to see the actual condition and characteristics of the item. The product configurator of the company Design Italian Shoes [40], thanks to its 3D model animation, allows the buyer to perceive the displayed article as tangible. Due to this function, the visual sense processes a concrete image of the item to be purchased.

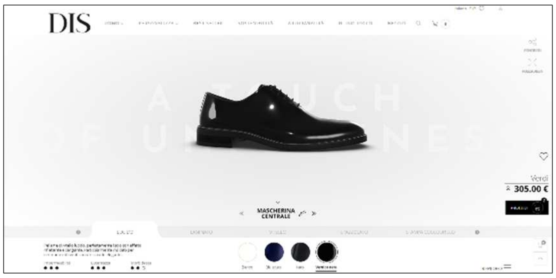


Figure 2: Shoe configurator [40]. Screenshot from https://www.designitalianshoes.com/apps/zakeke/c/verdi-custom?sku=120A00020106XX_353535, with permission from DIS - Design Italian Shoes.

3.5. Realism

3.5.1. Definitions

“Realism refers to the extent to which one believes the virtual environment is real [41] – the degree to which one feels the virtual space represents the actual physical space” [29: 1055]. It also “refers to the perceived correspondence between a technology-mediated experience and a similar experience not mediated by technology [42]” [43: 3].

3.5.2. Examples

When designing a building, architects take advantage of 3D virtual environments, software with a high degree of realism, in which physical spaces that do not yet exist can be created and experienced [29].

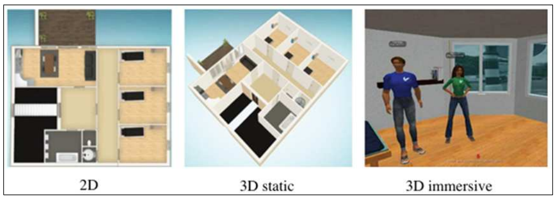


Figure 3: Three visual presentations of an apartment [29: 1058]. Reprinted from Computers in Human Behavior, 29, S. van der Land, A. P. Schouten, F. Feldberg, B. van den Hooff, M. Huysman, Lost in space? Cognitive fit and cognitive load in 3D virtual environments, 1054-1064, Copyright (2013), with permission from Elsevier.

Van der Land et al. [29] state that when the degree of realism of a virtual environment reaches the affinity of the real world, the user is assisted by the technology in the process of making decisions and understanding their concrete problem.

The 'Street View' feature of Google Maps provides a realistic visualization of the route or location that the individual is searching for. Thanks to this feature, the user can get an idea of what he or she will be facing in the real future, helping to increase safety. Manufacturers of in-car navigation assistance devices offer reality visualization features that help drivers. For example, they provide high-quality images when approaching a motorway exit or give advice on which lane to take in situations with complex arterial roads [44].

3.6. Media richness

3.6.1. Definitions

Media richness regards the "extent to which interface facilitates communication [45]" [9: 5]. "Media can be characterized as high or low in 'richness' based on their capacity to facilitate shared meaning" [46: 358]. According to Daft et al. [46], this factor is based on a combination of four criteria.

1. Feedback: instant feedback that allows for specific questions and corrections during the interaction;
2. Multiple cues: a range of cues from which a message can be composed, including physical presence, body gestures, graphic symbols, numbers, and words;
3. Language variety: the range of meanings that can be conveyed by language symbols. For example: numbers convey greater precision than natural language, while the latter helps to convey understanding of a wider range of ideas and concepts [46, 47];
4. Personal focus: a message is conveyed more fully when its communication evokes personal feelings and emotions.

3.6.2. Examples

The following four media channels (unaddressed documents, addressed documents, telephone, and face-to-face) are based on the hierarchy proposed by Daft et al. [46] and are presented in ascending order of media richness, from low to high.

Examples of unaddressed documents are fliers, bulletins, and standard quantitative reports. They mainly use messages consisting of three signals: graphic symbols, words, and numbers. They are useful for communicating quantifiable information, but they lack the ability to convey articulated ideas and concepts. They also lack personalization because they are delivered to a large audience and it is impossible to get feedback.

Addressed documents, such as letters, express written information. They are suitable for presenting personalized content tailored to the individual, thus creating a psychological connection with the reader. Despite this positive criterion, the feedback is still slow.

The telephone call has the ability to provide feedback quickly. Interlocutors rely on their own voice, tone, and language skills to convey messages. This type of channel, which uses natural language communication in real time, is quite rich. A rich communication tool facilitates intuition and allows for rapid understanding of the content presented.

The 'face-to-face' conversation is considered to be the form of interaction with the highest level of this factor. It allows for a rapid exchange of feedback and the creation of a direct connection with the interlocutor. The sender sends multiple signals, making the message comprehensive; the receiver develops empathy more quickly and the content may evoke emotions in him/her.

3.7. Avatar similarity

3.7.1. Definitions

The avatar is everyone's technological alter ego and the digital representation that each user chooses to identify themselves in an online community.

Avatar similarity is defined "as the perceived similarity between the avatar' s physical appearance and the user's physical appearance; the extent of an avatar's similarity is regarded as the degree of reflection of self-concept" [48: 715]. The construction of this virtual element involves the creation of a bond with it, as it is seen as a representation of oneself within the simulated environment [48, 49, 50].

3.7.2. Examples

Suh et al. [48] highlight methods that allow the generation of an avatar in its likeness. We begin with the production of an avatar that incorporates the feature of facial similarity. Two production methods are presented: a three-dimensional facial scan and the selection of a generic face among those proposed, with modifications of the features considered significant. A 3D scanner makes it possible to obtain the precise shape of the face; during the scan, several high-resolution digital photographs are taken, resulting in a virtual face. The subject will undoubtedly perceive a closer resemblance to the first process described. In a very similar way, body similarity has been achieved using three-dimensional body scanning and the use of the body mass index. Although it is clear that the first technique gives a better result, it has a number of limitations, such as the cost of the machine and a difficult detection procedure. The second method requires the subject to enter values such as height, weight, and age, and to select the body shape that most resembles their own from those presented. One hundred body shapes are included in the database, and the system processes the most suitable according to the data received; the user can then model the features that he or she considers most important.

3.8. Functional control

3.8.1. Definitions

"Functional control enables consumers to explore and experience different features and functions of products" [51: 111]. Functional control regards the "manipulation of product functionality to understand how a product works [51]" [9: 5].

3.8.2. Examples

The functional control enabled by software such as Shockwave allows consumers to test different attributes of products on their computers [51].

An obvious demonstration is online interaction with some product configurators, such as those for electronic watches [51]. As the user adds new features, such as buttons to the item, the cyber product is updated, and its virtual operation mirrors real behavior. Reactions, such as the sound produced by pressing a button, can be reproduced by the sound emitted by clicking on the virtual representation of that particular item.

It is worth mentioning PhET's website for creating simulated electrical circuits, as it clearly implements this factor. By simply clicking on the various objects presented in the interface, it is possible to create an electrical circuit and simulate it so that the user can understand how it works in order to implement it in practice.

3.9. Visual control

3.9.1. Definitions

"Visual control enables consumers to manipulate Web product images, to view products from various angles and distances" [51: 111]. It regards the "manipulation of product images to understand product looks by moving, rotating and zooming [51]" [9: 5].

3.9.2. Examples

Functionalities such as panning, rotating and zooming a virtual product image allow the user to enjoy visual control. The latter, implemented in software such as QuickTime and Flash, allows consumers to manage visual operations using the mouse and keyboard [51].

This factor is particularly noticeable when interacting with the graphical representations in online configurators. Ikea's virtual environment allows for an extremely detailed visual experience for the consumer [52]. Once the room has been configured, it is possible to move around the room using the mouse to view any details.

3.10. Interaction richness

3.10.1. Definitions

Interaction richness is the "possibility of interaction with products and seller [53]" [9: 5]. It is also useful to quote the definition of ComputerLanguage.com [54], where 'rich interaction' is described as a beneficial and pleasant user experience when using an electronic device. It is worth pointing out that the concept of 'rich' is constantly evolving according to the technologies available; in fact, today it means software with a very intuitive and easy-to-use interface, while in the future it will be through tools such as automatic assistance and voice recognition.

3.10.2. Examples

The experiment by Jahng et al. [53] provides two examples with different levels of interaction richness for each of the dimensions mentioned above. The richness of interaction with the salesperson can be achieved through two means of communication: email and an interface implementing real-time two-way audio and one-way video. It is clear that the second interaction is richer because it makes the product presentation more effective and the communication support to the customer more direct.

On the other hand, the richness of the interaction with the product can take the form of a simple static image associated with a textual description of the specifications, or an interactive three-dimensional test of the product features accompanied by an explanation.

The effectiveness of the latter is obvious, as it allows better manipulation to understand the features and overall functioning of the product. Banking service sites such as Fineco offer a feature such as 'live chat'. Live chat is a type of customer service software that allows visitors to interact with operators in real time and through a chat window [55]. It is clear that effective customer service is provided through interactive interfaces that allow the user to engage in profitable communication before and after purchase.

3.11. Vividness

3.11.1. Definitions

"Vividness means the representational richness of a mediated environment as defined by its formal features, that is, the way in which an environment presents information to the senses" [24: 75]. "Vividness refers to the abundance of information that an external environment (e.g., AR app) gives to the human senses, of which perceptual breadth and perceptual depth are the two primary variables [24]" [11: 4]. "Perceptual breadth refers to the extent to which an individual utilizes multiple sensory modalities to perceive information simultaneously" [11: 4]. "Perceptual depth is defined as the measure of both the quantity and quality of information received by the senses" [11: 4].

3.11.2. Examples

"The quality of the media directly impacts perceptual depth, and more advanced technology (e.g., AR) generally has better perceptual depth performance" [11: 4]. Vonkeman et al. [18] present interesting examples of vividness while investigating the effect of different product (in the specific case sunglasses) display formats, namely: static images, 360-degree rotation tool and virtual mirror.

Figure 4: Virtual mirror [18: 1046]. Reprinted from Information & Management, 54, C. Vonkeman, T. Verhagen, W. van Dolen, Role of local presence in online impulse buying, 1038-1048, Copyright (2017), with permission from Elsevier.

The static images are captured by a simple camera and are noninteractive, whereas the 360-degree rotation tool allows the displays to be rotated by clicking and dragging the mouse. A webcam captures the participants' faces and transfers them to the screen via a 'virtual mirror' application by placing the product on it. The three presentation modes differ in terms of vividness, particularly in terms of depth, i.e., the quality of sensory stimulation provided. Static images provide fixed visual input and elicit the lowest level, while the 360-degree rotation tool and the 'virtual mirror' provide higher levels through real-time manipulation. The latter, in particular, through the captured image of the face, provides a higher quality of representation, and thus a higher level of vividness.

4. Discussion and conclusion

This paper continues the line of research regarding visualization in configurators [e.g., 8, 56]. In particular, it contributes to that line by distinguishing dimensions for describing and evaluating product visualization modes by drawing on literature from e-commerce and customer experience. Based on literature on e-commerce and customer experience we identified a list of 11 variables that can be used as a framework for describing and evaluating all product visualization modes in configurators. Each variable of the framework is defined and exemplified by practical examples.

While this study provides a starting point for evaluating product visualization modes in configurators, it predominantly draws from the e-commerce domain. Consequently, certain aspects specifically related to the VR/AR/MR domain may not be fully captured. For example, in immersive environments, flow, motion sickness, dizziness due to avatar similarity, and the uncanny valley effect may arise. Future research could integrate insights from related disciplines, particularly those focusing on immersive technologies, to enrich the framework or to identify relationships among the framework's variables and other relevant outcome variables.

Additionally, we acknowledge that some of the proposed 11 variables may be correlated. However, at this stage, there is no empirical or theoretical basis to reduce them to a smaller set. We believe it is beneficial to present all variables and address potential consolidation in future research.

Regarding the practical contributions of this study, the identified dimensions can serve as valuable tools for managers who need to select visualization alternatives and for designers responsible for developing product visualization interfaces in configurators. The proposed framework, when applied to describe and compare different visualization modes, allows the presentation and comparison of the unique features, advantages, and disadvantages of the modes. This detailed description is effective for managers and designers who are choosing the most suitable product visualization mode for their specific context, providing crucial support during the evaluation process. Additionally, this framework allows one to systematically compare these visualization modes, simplifying the decision-making process.

However, the study did not apply these evaluation dimensions to the various visualization modes in configurators, which would further benefit both managers and designers. Therefore, the next step involves characterizing these visualization modes on the basis of the 11 dimensions that comprise the proposed framework. To achieve this, it is necessary to develop appropriate measurement metrics. While some metrics have already been proposed in the literature, others will need to be developed. In any case, it is crucial to adapt these metrics and test their effectiveness within the context of configurators.

Finally, new technologies come with limitations in costs and benefits. High development and implementation costs, including hardware and software, can be significant barriers. User accessibility and the need for specialized equipment could also limit adoption. For products not requiring spatial interaction or immersive experiences, traditional visualizations can still provide a satisfactory configuration experience at a lower cost. Future research should explore cost-benefit analyses to identify scenarios where advanced visualization modes offer distinct advantages over conventional approaches.

Acknowledgements

The authors thank the University of Padova for funding projects DOR2334949 and DTG SID 2024.

References

[1] W. Di, N. Sundaresan, R. Piramuthu, A. Bhardwaj, Is a picture really worth a thousand words? - on the role of images in e-commerce, Proceedings of the 7th ACM international conference on Web search and data mining, New York, New York, USA, Association for Computing Machinery, 2014, pp. 633–642. doi: 10.1145/2556195.2556226.

[2] S. Christianson, 100 diagrams that changed the world, Penguin, New York, NY, 2012.

[3] J. Opiła, Role of visualization in a knowledge transfer process, Business Systems Research Journal 10 (2019) 164-179. doi: 10.2478/bsrj-2019-0012.

[4] J. van Biljon, K.-M. Osei-Bryson, The communicative power of knowledge visualizations in mobilizing information and communication technology research, Information Technology for Development 26 (2020) 637-652. doi: 10.1080/02681102.2020.1821954.

[5] J. Gans, A. Nagaraj, What is Apple's vision pro really for, Harvard Business Review June 14 (2023).

[6] SkyQuest Technology, Global AR/VR/MR market, 2022. URL: https://www.skyquestt.com/report/ar-vr-mr-market.

[7] H. Ebbesen, C. Machholdt, Digital reality changes everything: step into the future, Deloitte Development LLC (2019).

[8] E. Sandrin, C. Forza, Visualization in configurators: reflections for future research, in J. M. Horcas, J. Á. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proceedings of the 25th International Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6–7, CEUR Workshop Proceedings, 2023, pp. 8-11.

[9] W. Zeng, A. Richardson, Beyond 2D product presentation in e-commerce: a literature review, ACIS 2017 Proceedings: 35, 2017.

[10] C. Flavián, S. Ibáñez-Sánchez, C. Orús, The impact of virtual, augmented and mixed reality technologies on the customer experience, Journal of Business Research 100 (2019) 547-560. doi: 10.1016/j.jbusres.2018.10.050.

[11] W.-C. Hsu, M.-H. Lee, K.-W. Zheng, From virtual to reality: the power of augmented reality in triggering impulsive purchases, Journal of Retailing and Consumer Services 76 (2024) 1-13. doi: 10.1016/j.jretconser.2023.103604.

[12] D. Ihde, Technology and the lifeworld: From garden to earth, Indiana University Press, Bloomington, IN, 1990.

[13] F. Biocca, The cyborg's dilemma: progressive embodiment in virtual environments, Journal of Computer-Mediated Communication 3 (1997). doi: 10.1111/j.1083-6101.1997.tb00070.x.

[14] I. Tussyadiah, Expectation of travel experiences with wearable computing devices, in Z. Xiang, I. Tussyadiah (Eds.), Information and Communication Technologies in Tourism, Proceedings of the International Conference in Dublin, January 21-24, Springer, Cham, Switzerland, 2014, pp. 539-552. doi: 10.1007/978-3-319-03973-2_39.

[15] D.-H. Shin, The role of affordance in the experience of virtual reality learning: technological and affective affordances in virtual reality, Telematics and Informatics 34 (2017) 1826-1836. doi: 10.1016/j.tele.2017.05.013.

[16] M. Rudd, K. D. Vohs, J. Aaker, Awe expands people's perception of time, alters decision making, and enhances well-being, Psychological Science 23 (2012) 1130-1136. doi: 10.1177/0956797612438731.

[17] G. Giordano, Lenti intraoculari, 2018. URL: https://oculistagiordano.it/lenti-intraoculari/.

[18] C. Vonkeman, T. Verhagen, W. van Dolen, Role of local presence in online impulse buying, Information & Management 54 (2017) 1038-1048. doi: 10.1016/j.im.2017.02.008.

[19] J. A. Waterworth, E. L. Waterworth, G. Riva, F. Mantovani, Presence: form, content and consciousness, in M. Lombard, F. Biocca, J. Freeman, W. Ijsselsteijn, R. J. Schaevitz (Eds.), Immersed in Media: Telepresence Theory, Measurement & Technology, Springer, Cham, Switzerland, 2015, pp. 35-58. doi: 10.1007/978-3-319-10190-3_3.

[20] C. A. Thornson, B. F. Goldiez, H. Le, Predicting presence: constructing the tendency toward presence inventory, International Journal of Human-Computer Studies 67 (2009) 62-78. doi: 10.1016/j.ijhcs.2008.08.006.

[21] C. Coelho, J. Tichon, T. J. Hine, G. Wallis, G. Riva, Media presence and inner presence: the sense of presence in virtual reality technologies, in G. Riva, M. T. Anguera, B. K. Wiederhold, F. Mantovani

(Eds.), From Communication to Presence: Cognition, Emotions and Culture Towards the Ultimate Communicative Experience, IOS Press, Amsterdam, Netherlands, 2006, pp. 25-45.
[22] M. Slater, Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments, Philosophical Transactions of the Royal Society B: Biological Sciences 364 (2009) 3549-3557. doi: 10.1098/rstb.2009.0138.
[23] S. Ang, J. Quarles, Reduction of cybersickness in head mounted displays use: a systematic review and taxonomy of current strategies, Frontiers in Virtual Reality 4 (2023). doi: 10.3389/frvir.2023.1027552.
[24] J. Steuer, Defining virtual reality: dimensions determining telepresence, Journal of Communication 42 (1992) 73-93. doi: 10.1111/j.1460-2466.1992.tb00812.x.
[25] M. Carrozzino, M. Bergamasco, Beyond virtual museums: experiencing immersive virtual reality in real museums, Journal of Cultural Heritage 11 (2010) 452-458. doi: 10.1016/j.culher.2010.04.001.
[26] M. A. Muhanna, Virtual reality and the CAVE: taxonomy, interaction challenges and research directions, Journal of King Saud University - Computer and Information Sciences 27 (2015) 344-361. doi: 10.1016/j.jksuci.2014.03.023.
[27] D. Sohn, Anatomy of interaction experience: distinguishing sensory, semantic, and behavioral dimensions of interactivity, New Media & Society 13 (2011) 1320-1335. doi: 10.1177/1461444811405806.
[28] A. Mollen, H. Wilson, Engagement, telepresence and interactivity in online consumer experience: reconciling scholastic and managerial perspectives, Journal of Business Research 63 (2010) 919-925. doi: 10.1016/j.jbusres.2009.05.014.
[29] S. van der Land, A. P. Schouten, F. Feldberg, B. van den Hooff, M. Huysman, Lost in space? Cognitive fit and cognitive load in 3D virtual environments, Computers in Human Behavior 29 (2013) 1054-1064. doi: 10.1016/j.chb.2012.09.006.
[30] S. J. McMillan, J.-S. Hwang, Measures of perceived interactivity: an exploration of the role of direction of communication, user control, and time in shaping perceptions of interactivity, Journal of Advertising 31 (2002) 29-42. doi: 10.1080/00913367.2002.10673674.
[31] M. L. Heilig, Sensorama simulator, 1962.
[32] K. Grayson, R. Martinec, Consumer perceptions of iconicity and indexicality and their influence on assessments of authentic market offerings, Journal of Consumer Research 31 (2004) 296-312. doi: 10.1086/422109.
[33] E. M. Bruner, Abraham Lincoln as authentic reproduction: a critique of postmodernism, American Anthropologist 96 (1994) 397-415.
[34] J. Huntington, Philip K. Dick: authenticity and insincerity, Science Fiction Studies 15 (1988) 152-160.
[35] M. Crang, Magic kingdom or a quixotic quest for authenticity?, Annals of Tourism Research 23 (1996) 415-431. doi: 10.1016/0160-7383(95)00070-4.
[36] R. A. Peterson, Creating country music: fabricating authenticity, University of Chicago Press, Chicago, IL, 2013.
[37] R. Algharabat, C. Dennis, Using authentic 3D product visualisation for an electrical online retailer, Journal of Customer Behaviour 9 (2010) 97-115. doi: 10.1362/147539210X511326.
[38] T. C. O'Guinn, Touching greatness: the central midwest Barry Manilow fan club, in R. Belk (Ed.), Highways and Buyways: Naturalistic Research from the Consumer Behavior Odyssey, Association for Consumer Research, Provo, UT, 1991, pp. 102-111.
[39] J. A. Costa, G. J. Bamossy, Culture and the marketing of culture: the museum retail context, in J. A. Costa, G. J. Bamossy (Eds.), Marketing in a multicultural world: Ethnicity, nationalism, and cultural identity, SAGE, Thousand Oaks, CA, 1995, pp. 299-328.
[40] DIS - Design Italian Shoes, Verdi custom, 2024. URL: https://www.designitalianshoes.com/en/apps/zakeke/c/verdi-custom?sku=120A00020106XX_353535.
[41] A. Davis, J. Murphy, D. Owens, D. Khazanchi, I. Zigurs, Avatars, people, and virtual worlds: foundations for research in metaverses, Journal of the Association for Information Systems 10 (2009) 90-117. doi: 10.17705/1jais.00183.
[42] M. Lombard, M. T. Jones, Defining presence, in M. Lombard, F. Biocca, J. Freeman, W. Ijsselsteijn, R. J. Schaevitz (Eds.), Immersed in Media: Telepresence Theory, Measurement & Technology, Springer, Cham, Switzerland, 2015, pp. 13-34. doi: 10.1007/978-3-319-10190-3_2.
[43] M. Daassi, S. Debbabi, Intention to reuse AR-based apps: the combined role of the sense of immersion, product presence and perceived realism, Information & Management 58 (2021) 1-12. doi: 10.1016/j.im.2021.103453.
[44] P. Fröhlich, M. Baldauf, M. Hagen, S. Suette, D. Schabus, A. L. Kun, Investigating safety services on the motorway: the role of realistic visualization, in M. Tscheligi, M. Kranz, G. Weinberg, A. Meschtscherjakov, M. Murer, D. Wilfinger (Eds.), Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, 30 November - 2 December, Salzburg, Austria, Association for Computing Machinery, New York, NY, 2011, pp. 143–150. doi: 10.1145/2381416.2381440.
[45] H. Kim, K.-S. Suh, U.-K. Lee, Effects of collaborative online shopping on shopping experience through social and relational perspectives, Information & Management 50 (2013) 169-180. doi: 10.1016/j.im.2013.02.003.
[46] R. L. Daft, R. H. Lengel, L. K. Trevino, Message equivocality, media selection, and manager performance: implications for information systems, MIS Quarterly 11 (1987) 355-366. doi: 10.2307/248682.

[47] R. L. Daft, J. C. Wiginton, Language and organization, Academy of Management Review 4 (1979) 179-191. doi: 10.5465/amr.1979.4289017.

[48] K.-S. Suh, H. Kim, E. K. Suh, What if your avatar looks like you? Dual-congruity perspectives for avatar use, MIS Quarterly 35 (2011) 711-729. doi: 10.2307/23042805.

[49] T. Jordan, Cyberpower: The Culture and Politics of the Internet and Cyberspace, Routledge, London, UK, 1999.

[50] Y. B. Kafai, D. A. Fields, M. S. Cook, Your second selves: player-designed avatars, Games and Culture 5 (2010) 23-42. doi: 10.1177/1555412009351260.

[51] Z. Jiang, I. Benbasat, Virtual product experience: effects of visual and functional control of products on perceived diagnosticity and flow in electronic shopping, Journal of Management Information Systems 21 (2004) 111-147. doi: 10.1080/07421222.2004.11045817.

[52] Ikea, BESTÅ planner, 2024. URL: https://www.ikea.com/addon-app/storageone/besta/web/latest/at/en/#/planner.

[53] J. Jahng, H. Jain, K. Ramamurthy, Effects of interaction richness on consumer attitudes and behavioral intentions in e-commerce: some experimental results, European Journal of Information Systems 16 (2007) 254-269. doi: 10.1057/palgrave.ejis.3000665.

[54] ComputerLanguage.com, Definition: rich interaction, 2024. URL: https://www.computerlanguage.com/results.php?definition=rich+interaction.

[55] G. McLean, K. Osei-Frimpong, Chat now… Examining the variables influencing the use of online live chat, Technological Forecasting and Social Change 146 (2019) 55-67. doi: 10.1016/j.techfore.2019.05.017.

[56] L. Hvam, K. Ladeby, An approach for the development of visual configuration systems, Computers & Industrial Engineering 53 (2007) 401-419. doi: 10.1016/j.cie.2007.05.004.

Using Answer Set Programming for Assigning Tasks to Computing Nodes

Franz Wotawa[1,*]

[1]Graz University of Technology (TU Graz), Institute of Software Technology, Inffeldgasse 16b/2, A-8010 Graz, Austria

Abstract

Allocating tasks to computing nodes in a network is an important configuration problem. In the case of fail-safe networks, such configuration must be changed during operation if a computing node fails. Hence, a fast configuration is required. In this paper, we formulate a tasks-to-computing-nodes assignment problem and its constraints using answer set programming. We performed an initial experimental evaluation utilizing several smaller to mid-size problem instances to show whether logic reasoning based on answer set programming is feasible for practical applications. We discovered that reasoning is fast if a solution exists but not when there is no solution. Further constraints help to decide that a problem instance is unsolvable early in the search, which improves the outcome.

Keywords

Configuring computing nodes, ASP models for configuration, Experimental analysis

1. Introduction

There has been plenty of work in the area of configuration and recommender systems, including service configuration [1], governance systems [2], or product configuration [3] to refer to recent work. However, answer set programming for representing models used for configuration and reasoning to obtain valid configuration has recently gained more attention, e.g., see [3, 4, 5]. In this paper, we contribute to this research direction and introduce a model and an evaluation for configuring networks comprising computing nodes for executing pre-defined tasks. The underlying problem is related to scheduling and shift designs [6].

The main motivation behind our work comes from practical applications, where, for example, tasks, i.e., programs, have to be deployed on computing nodes in a network. Although this problem can be seen as a static one that must only be solved before the operation, we may require to re-configure such a task assignment during operation whenever a computing node fails (see, e.g., [7]). Such re-configuration tasks have to be carried out under time restrictions. To evaluate whether modern reasoning methods like answer set programming can be used for this task, we conduct an experimental evaluation. This evaluation comprises several instances of the corresponding task to computing node assignment problems considering various sizes of nodes and tasks. The evaluation utilizes the answer set programming solver `clingo` [8]. The question we want to answer is regarding the approach's limitations regarding the problem size. Can we use answer set programming (and in particular `clingo`) to provide task assignments fast enough to be used during operation?

We structure this paper as follows. First, we introduce the underlying configuration problem and `clingo` implementation. Afterward, we discuss the experimental evaluation, i.e., the basic setup and the results obtained. Finally, we conclude the paper.

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.

wotawa@ist.tugraz.at (F. Wotawa)

https://www.tugraz.at/ (F. Wotawa)

0000-0002-0462-2283 (F. Wotawa)

2. Problem description

We start defining the task to node assignment problem. We assume we have k computing nodes $n_1, \ldots, n_k$ and n tasks $t_1, \ldots, t_n$ to be assigned to the nodes. For each node n_i, we know the maximum number of tasks $c(n_i)$ it can hold and the available memory $m(n_i)$. For each task t_j, we know its memory consumption $m(t_j)$. From this knowledge, we obtain several constraints an assignment must fulfill to be valid. In the following, we formalize these constraints assuming that the function $assigned(n_i)$ returns a set of tasks that is assigned to a node n_i

First, the required memory from the tasks assigned to a node shall never exceed the available memory of this node. These constraints can be formalized as follows:

$$\forall i \in \{1, \ldots, k\} : \left(\sum_{t_j \in assigned(n_i)} m(t_j) \leq m(n_i) \right) \quad (1)$$

Second, the number of tasks assigned to a node shall never exceed the maximum number of tasks the node can hold, i.e., formally, we write:

$$\forall i \in \{1, \ldots, k\} : (|assigned(n_i)| \leq c(n_i)) \quad (2)$$

A solution to the tasks to computing nodes assignment problem is an assignment of all tasks to all nodes such that $\forall j \in \{1, \ldots, k\} : \exists i \in \{1, \ldots, n\} : t_j \in assigned(n_i)$, there is no tasks assigned to two different nodes, i.e., $\forall i, j \in \{1, \ldots, k\}, i \neq j : assigned(n_i) \cap assigned(n_j) = \emptyset$, and all constraints are fulfilled. Such an assignment is a valid one and may not exist. For example, if the number of tasks exceeds the number of free slots of the nodes or if the required memory for the tasks is not provided, there is no solution. Hence, we have two necessary conditions that must hold for obtaining a solution, i.e.:

$$n \leq \sum_{i=1}^{k} c(n_i) \tag{3}$$

$$\sum_{j=1}^{n} m(t_j) \leq \sum_{i=1}^{k} m(n_i) \tag{4}$$

It is worth noting that similar problems have additional constraints, e.g., stating that tasks need to be together in the same computing node. We may also introduce optimality criteria like preferring solutions requiring the least amount of computing nodes. Furthermore, we may also consider variants of the problem, i.e., reconfiguration of assignments. In the context of this paper, we do not tackle such extensions. We solely focus on answering the question regarding the applicability of the answer set program to solve the problem within a reasonable amount of time.

After outlining the problem in general, we present a solution using answer set programming where we rely on the syntax of the `clingo` solver[1] [8], which is similar to the Prolog language. Due to space restrictions, we do not give an introduction to answer set programming (ASP). Instead, we refer to introductory literature into ASP, e.g., [9].

A `clingo` model for the node assignment problem comprises three parts. First, we define the number of computing nodes and tasks and their capacities and requirements. For every node, e.g., `n2`, we use three facts, where the predicate `tcapacity` denotes the maximum number of tasks, and `mcapacity` the maximum available memory for the given task.:

```
node(n2). tcapacity(n2,1). mcapacity(n2,20).
```

For each task, e.g., `t1`, we add two facts, where the predicate `memory` is for defining the required memory of the given task to a pre-defined value:

```
task(t1). memory(t1,30).
```

Second, we generate all potential solutions. For this purpose, we introduce a predicate for a node assignment of a task. Let us call this predicate `select` that takes a task `T` as the first parameter and node `N` as the second. In `clingo`, the generate part for the node assignment problem is given as follows:

```
{ select(T,N) : node(N) } = 1 :- task(T).
```

This rule generates a grounded predicate that assigns tasks to each computing node. Obviously, not all assignments are correct when considering the constraints. Hence, in the last part, we formalize the first two constraints of the general problem, i.e., Equations 1 and 2, but not the other Equations 3 and 4. For this purpose, we introduce a predicate `nrTasksAssigned` that holds the number of tasks that are assigned to a particular node and a predicate `memRequired` that holds the required memory for a node considering the assigned tasks. The information regarding the predicates can be obtained from the selected task for a node (`select` predicate) and the memory required for a task. For the latter, we introduce the predicate `memory`. The predicate `nrTasksAssigned` can be formalized in `clingo` as follows:

```
nrTasksAssigned(N,M) :-
  M = #count {T:select(T,N)},
  node(N).
```

Similar, we can define the `memRequired` predicate:

```
memRequired(N,M) :-
  M = #sum {NM:select(T,N), memory(T,NM)},
  node(N).
```

Using these predicates, we can formulate the constraints:

```
:- nrTasksAssigned(N,M), tcapacity(N,C), M>C.
:- memRequired(N,M), mcapacity(N,C), M > C.
```

The first constraint states that it is impossible to assign more tasks to a node than the node can hold. The second constraint states that the memory requirements of all tasks should not exceed the memory capacity of the computing node.

3. Experimental evaluation

The following experimental evaluation aims to investigate the runtime behavior for finding one solution of the task to computation node assignment problem using the ASP solver `clingo`. In particular, we are interested in the number of nodes that can be handled requiring less than a fixed boundary of time, e.g., 0.01 or 0.1 seconds. In the following, we outline the experimental setup and present and discuss the obtained results.

[1] See https://potassco.org/about/

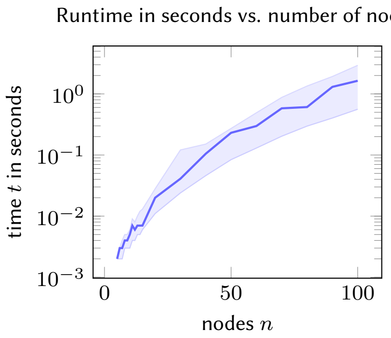


Figure 1: Solving runtime of consistent instances

Experimental setup: To develop several instances of the task assignment problem, we wrote a Java program for generating such instances automatically. We ranged the number of nodes from 5 to 100. The number of tasks for each instance was randomly chosen between the number of nodes and double the number of nodes. The capacity of each node was randomly set from 1,2, …, 10. The memory provided by each node was randomly chosen from 20, 40, 60,…, 200. The memory required by every task was randomly set from 10, 20, and 30. With this setup, we generated only satisfiable instances, i.e., problem instances where a solution exists. For a category of instances comprising n nodes, we called the instance generator 10 times. Finally, we received 200 different problem instances.

We conducted the experiments using an Apple MacBook Pro, with an Apple M1 CPU comprising 8 cores and 16 GB of main memory, running under macOS Sonoma Version 14.4.1. For computing solutions, we relied on `clingo` version 5.7.1 and applied the standard setup.

Experimental results: After generating the problem instances, we ran `clingo` to compute one solution, i.e., we ran `clingo` using the prompt `clingo -time-limit=10 -outf=2` for each file. Hence, we set a time limit of 10 seconds and obtained all results in JSON format. After analyzing the results for correctness, we summarized the outcome, i.e., the runtime for each category of a particular number of nodes. Figure 1 depicts the minimum, maximum, and average runtime for all 10 runs for each category.

We see that when using ASP solving utilizing `clingo` we can provide one solution even for larger instances of 100 computing nodes in a reasonable amount of time. However, when using the approach during operation, and especially for systems with harder requirements regarding answering time, e.g., real-time systems, a runtime of almost 3 seconds might not be feasible. We would like answers in less than 0.1 or 0.01 seconds for such systems, which can be achieved for 20 or 13 nodes, respectively.

Motivated by the results, we performed further experiments, considering problems that likely cannot be solved. Unfortunately, in this case, we often ran into reaching the solving time limit of 10 seconds, even starting with small examples only considering 5 computing nodes. Instances with more than 7 nodes that might be unsatisfiable always reach the 10-second limit. For those instances where unsatisfiability could be established, the runtime varies between 0.003 and 6.255 seconds. The latter was obtained for a problem instance comprising 7 computing nodes. Hence, unsatisfiable instances can hardly be identified when considering more computing nodes, which might also be an issue for practical applications.

We carried out another experiment to tackle the mentioned problem of potential unsatisfiability. We selected 3 problem instances for which we could not compute a result. Two instances had 7 nodes, and one had 15 nodes. For these problem instances, we added further constraints that cover Equations 3 and 4. For all three `clingo` files that correspond to the problem instances, we added the following code:

```
totalCapacity(C) :- C=#sum{T:tcapacity(N,T)}.
totalMemReq(C) :- C=#sum{M:memory(T,M)}.
totalMem(C) :- C=#sum{N:mcapacity(CN,N)}.
:- totalCapacity(C), C < 21.
:- totalMemReq(Ctask), totalMem(Cnode),
         Ctask > Cnode.
```

Note that the `21` in the above code represents the number of tasks[2]. We adapted this value for each instance and set it to `21`, `28`, and `60` respectively. When running `clingo` on the three files, we obtained an immediate response of unsatisfiability. In all cases, this response was less than 0.025 seconds. Hence, adding further constraints that allow distinguishing satisfiable from unsatisfiable cases as early as possible solves the problem.

In summary, `clingo` allows for fast computation of solutions if they exist. The reasoning for the mentioned tasks-to-computing-nodes assignment problem is fast enough for at least smaller examples to ensure a timely response. However, whether this is good enough depends on the application domain. The challenges we obtained in the case of unsatisfiability can be solved by setting a time limit for `clingo` and introducing additional constraints. Results from this case study may also apply to other configuration problems.

Threats to validity: There are many threats to validity worth mentioning. The experimental evaluation is

[2]The number of tasks for a particular problem instance can also be obtained using `clingo`. The command `#count{T:task(T)}` delivers this number. However, we set the number manually for the three experiments.

limited in the number of problem instances. There might be satisfiable instances that may take longer than reported for a given number of computing nodes. However, we do not expect a very large deviation from the results. Furthermore, we only considered one rather simple configuration problem. In case of more complex problems, we expect different runtimes. Nevertheless, the effect of adding constraints to determine unsatisfiability as early as possible should still be visible. This might also hold for the observation that unsatisfiability might be hard to identify and, therefore, require more computing time. We carried out all experiments using `clingo`'s standard-setting. There might be differences to observe when changing parameters and setup. There might also be differences when considering other versions of `clingo`, the hardware, or the operating system. Finally, the representation of the problem in `clingo` might also influence the performance.

4. Conclusions

In this paper, we used the configuration problem of assigning tasks to computing nodes to answer whether answer-set programming is feasible for practical applications. For smaller problem instances, answer set programming might be feasible, providing a fast response within a fraction of a second. For larger instances, we may not be able to provide a solution within a reasonable answer time. Furthermore, we identified a challenge, i.e., the extended runtime required for providing an answer in case of unsatisfiability, and a solution, i.e., the effect of additional constraints on reducing the runtime. In future research, we want to extend the configuration problem to capture the case of task assignments for computing networks at runtime. For this purpose, we want to formulate a corresponding re-configuration problem. Furthermore, we want to extend the experimental evaluation using more example instances and additional constraints and consider computing optimal solutions concerning a given optimization criteria, e.g., using the least number of computing nodes.

References

[1] E. M. Strøm, T. M. Münsberg, L. Hvam, Identifying potential applications of service configuration systems in a logistics company, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proc. 25th Intl. Workshop on Configuration, Málaga, Spain, September 6-7, 2023, CEUR-WS.org, 2023, pp. 60–66. URL: https://ceur-ws.org/Vol-3509/paper9.pdf.

[2] S. Muñoz-Hermoso, D. Benavides, F. J. D. Mayo, Multi-level configuration in smart governance systems, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proc. 25th Intl. Workshop on Configuration, Málaga, Spain, September 6-7, 2023, CEUR-WS.org, 2023, pp. 67–74. URL: https://ceur-ws.org/Vol-3509/paper10.pdf.

[3] R. Comploi-Taupe, G. Friedrich, T. Niestroj, Dynamic aggregates in expressive ASP heuristics for configuration problems, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proc. 25th Intl. Workshop on Configuration, Málaga, Spain, September 6-7, 2023, CEUR-WS.org, 2023, pp. 75–84. URL: https://ceur-ws.org/Vol-3509/paper11.pdf.

[4] N. Rühling, T. Schaub, T. Stolzmann, Towards a formalization of configuration problems for asp-based reasoning: Preliminary report, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proc. 25th Intl. Workshop on Configuration, Málaga, Spain, September 6-7, 2023, CEUR-WS.org, 2023, pp. 85–94. URL: https://ceur-ws.org/Vol-3509/paper12.pdf.

[5] R. Comploi-Taupe, A. A. Falkner, S. Hahn, T. Schaub, G. Schenner, Interactive configuration with ASP multi-shot solving, in: J. M. Horcas, J. A. Galindo, R. Comploi-Taupe, L. Fuentes (Eds.), Proc. 25th Intl. Workshop on Configuration, Málaga, Spain, September 6-7, 2023, CEUR-WS.org, 2023, pp. 95–103. URL: https://ceur-ws.org/Vol-3509/paper13.pdf.

[6] M. Abseher, M. Gebser, N. Musliu, T. Schaub, S. Woltran, Shift design with answer set programming, in: F. Calimeri, G. Ianni, M. Truszczynski (Eds.), Proc. 13th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning, 2015, Lexington, KY, USA, September 27-30, 2015., volume 9345 of *LNCS*, Springer, 2015, pp. 32–39. URL: https://doi.org/10.1007/978-3-319-23264-5_4. doi:10.1007/978-3-319-23264-5_4.

[7] A. Ballesteros, M. Barranco, J. Proenza, L. Almeida, F. Pozo, P. Palmer-Rodríguez, An infrastructure for enabling dynamic fault tolerance in highly-reliable adaptive distributed embedded systems based on switched ethernet, Sensors 22 (2022) 7099. URL: https://doi.org/10.3390/s22187099. doi:10.3390/S22187099.

[8] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot asp solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82. doi:10.1017/S1471068418000054.

[9] T. Eiter, G. Ianni, T. Krennwallner, Answer set programming: A primer, in: S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M.-C. Rousset, R. A. Schmidt (Eds.), Reasoning Web. Semantic Technologies for Information Systems: 5th Intl. Summer School, Brixen-Bressanone, Italy, August 30 - September 4, 2009, pp. 40–110, Springer Berlin Heidelberg, . URL: https://doi.org/10.1007/978-3-642-03754-2_2. doi:10.1007/978-3-642-03754-2_2.

Responsible Configuration Using LLM-based Sustainability-Aware Explanations

Sebastian Lubos[1,*], Alexander Felfernig[1,*], Lothar Hotz[2], Thi Ngoc Trang Tran[1], Seda Polat-Erdeniz[1], Viet-Man Le[1], Damian Garber[1] and Merfat El-Mansi[1]

[1]*Institute of Software Technology, Graz University of Technology, Graz, Austria*
[2]*Hamburger Informatik Technologie-Center e.V., Hamburg, Germany*

Abstract
Configuration systems play an important role in achieving the sustainable development goals (SDGs) defined by the United Nations. As decision support systems, configurators help users to decide which components or features to include in or exclude from a configuration. An important task of configurators is the provision of explanations which help to achieve goals such as increasing configuration understandability, increasing a user's trust, and persuading users/customers to include specific configuration components. Our goal in this paper is to introduce the concept of „sustainability-aware explanations“ which can help to support the sustainable development goals. The type of explanations we propose in this context are somehow orthogonal to typical explanations used in industrial configuration environments. A major objective in this context is to follow a „less-is-more“ principle focusing on different aspects of the idea of „responsible configuration“ which refers to configuration techniques explicitly supporting the mentioned sustainability goals. We report the initial results of an evaluation that provide insights on potential impacts of the proposed explanations.

Keywords
Explanations, Sustainability, Green Configuration, Responsible Configuration, Configuration for Good, Nudging, Persuasion, Knowledge-based Configuration

1. Introduction

The 17 *sustainable development goals* (SDGs) defined by the United Nations (UN) provide a blueprint for peace and prosperity on our planet.[1] Examples of such goals are *good health and well-being* (e.g., in terms of fostering the consumption of healthy food), *responsible consumption and production* (e.g., in terms of reduced energy consumption), and *sustainable cities and communities* (e.g., in the context of tourism, avoiding negative environmental impacts and taking into account the local communities and cultural heritage) [1].

Knowledge-based configuration [2, 3, 4, 5] can be regarded as a core-technology of mass customization [6]. On the basis of configurators, users are enabled to design a product in an individualized fashion that fits their wishes and needs. In configuration settings, we can observe an ever-increasing demand for taking into account sustainability aspects [7, 8]. Following the basic definition of „*configuration*“ given by Sabin and Weigel [3], i.e., „***configuration is a special case of design activity where the configured artifact is assembled from a fixed set of well-defined component types and components are interacting in predefined ways***“, we define „***responsible configuration***“ as „***configuration which takes into account the United Nation's sustainable development goals***“.

In knowledge-based systems, explanations can be applied for different purposes [9]. First, so-called *why* explanations [10, 11, 12] focus on the aspect of mentioning the most relevant user requirements that lead to the determination of a specific configuration. Furthermore, *why not* explanations focus on supporting users in situations where no solution can be identified [13, 14, 15]. From the application point of view, explanations can be applied to achieve different goals [16].[2] Examples thereof are *efficiency* (reducing the time that is needed to complete a configuration task), *persuasiveness* (convincing users to change their component selection behavior), *transparency* (making the inclusion or exclusion of specific components transparent to the user), *trust* (increasing a user's confidence in the configuration system), *scrutability* (making it possible for the user to adapt the configurator behavior, e.g., in terms of the used component inclusion/exclusion strategy), and *satisfaction* (e.g., increasing the usability of a configuration system). These goals must be regarded as examples – for related details we refer to [11, 16, 17, 18].

In this paper, we focus on the *persuasion aspect* of explanations [19]. More precisely, we analyze possibilities to formulate explanations in such a way that users are nudged towards more sustainability-aware configuration decisions. Following a „less-is-more“ principle, we show how to formulate explanations following Cialdini's six principles of persuasion [20] (see Table 1).

Sustainability-aware explanations have to focus on argumentations including sustainability aspects. Our formulation of such explanations is based on *large language model* (LLM) prompts [21] which help to associate sustainable development goals with the mentioned persuasive principles. For example, in the context of car configuration, explanations could refer to the positive environmental aspects of purchasing smaller cars or on the advantages of electric vehicles compared to gasoline-driven ones.

Positive impacts of such sustainability-aware explanations can be, for example, higher-quality configuration decisions, a lower amount of unneeded components, and components with less negative

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
*Corresponding author.
slubos@ist.tugraz.at (S. Lubos); afelfern@ist.tugraz.at (A. Felfernig); lothar.hotz@uni-hamburg.de (L. Hotz); ttrang@ist.tugraz.at (T. N. T. Tran); spolater@ist.tugraz.at (S. Polat-Erdeniz); vietman.le@ist.tugraz.at (V. Le); dgarber@ist.tugraz.at (D. Garber); merfat.el-mansi@student.tugraz.at (M. El-Mansi)
0000-0002-5024-3786 (S. Lubos); 0000-0003-0108-3146 (A. Felfernig); 0000-0001-7370-7726 (L. Hotz); 0000-0002-3550-8352 (T. N. T. Tran); 0000-0001-5778-975X (V. Le)

[1]https://sdgs.un.org/goals

[2]The categorizations of [11, 16] have been developed in the context of recommender systems but can also be applied in configuration contexts.

Table 1
Cialdini's principles of persuasion [20].

principle	semantics
reciprocity	feeling of an obligation to give something back
scarcity	reduced item availability increases preparedness to purchase
authority	experts have an increased influence on users
commitment and consistency	users prefer to be consistent with their articulated preferences
liking	users like to comply with other users who are similar to themselves
social proof	users follow the opinions (of a representative set) of other users

environmental impacts [8]. From a commercial point of view, such explanations might appear – at least to some extent – counterproductive due to potential consequences in terms of decreasing turnovers. Thus, sustainability-aware explanations are often in contrast to explanations in mainstream configuration environments which focus on increasing sales rates in most of the cases.

The contributions of this paper are the following. First, we propose the concept of sustainability-aware explanations for configurations. Second, we provide reference examples of such explanations in the automotive domain. Third, we present initial results of a corresponding evaluation.

The remainder of this paper is organized as follows. In Section 2, we provide different examples of LLM-generated sustainability-aware explanations in the car configuration domain. Thereafter, we discuss initial results of a related evaluation (Section 3). In Section 4, we discuss threats to validity. Finally, we conclude the paper with Section 5.

2. Sustainability-Aware Explanations with LLMs

In the following, we discuss scenarios where sustainability-aware explanations can have an impact on user decisions. All scenarios are related to *car configuration* where users receive explanations of current configurations. The major goal of such explanations is to make users think about their current configuration settings and to potentially adapt their articulated preferences. Consequently, our explanations are not in the line of *why* or *why not* explanations but focus more on indicating potential alternatives to the current configuration, i.e., a kind of *why not choose something else* explanation. All example explanations in this paper have been generated on the basis of the LLM *ChatGPT 3.5*.[3]

Scenario 1: SUV vs. smaller car. The idea is to make persons (configurator users) who intend to purchase an SUV more aware of sustainability aspects of smaller cars. To support this, we have generated LLM-based explanations using the following (example) LLM prompt: *Assume the following scenario: person A wants to purchase a car and is interested in an SUV. Please provide persuasive explanations against purchasing an SUV following the six persuasion principles of Cialdini.* The resulting explanations are depicted in Table 2.

[3]https://chat.openai.com

Scenario 2: Long vs. standard range battery. The idea is to make configurator users interested in purchasing a car with a long-range battery aware of the sustainability aspects of standard-range batteries. To support such explanations, we have generated LLM-based explanations using the following LLM prompt: *person A wants to purchase an electric car and is interested in a long-range battery. Please provide persuasive arguments against purchasing a long range battery following the six persuasion principles of Cialdini.* The corresponding LLM-generated sustainability-aware explanations are depicted in Table 3.

Scenario 3: Car not needed in city center. Configurator users should think about the advantages of not having a car when living in the city center. We have generated related LLM-based persuasive explanations using the following LLM prompt: *person A who lives directly in the city center with various connections to public transportation wants to purchase a car. Please provide persuasive arguments against purchasing a car following the six persuasion principles of Cialdini.* The corresponding sustainability-aware explanations are depicted in Table 4.

Scenario 4: Less costly car due to financial situation. The idea is making configurator users with limited financial resources intending to purchase an expensive car to change their mind and purchase a less expensive car. To support such explanations, we have generated LLM-based explanations using the following (example) LLM prompt formulation: *person A with very limited financial resources and a family with three children wants to purchase an expensive car. Please provide persuasive arguments against purchasing an expensive car following the six persuasion principles of Cialdini.* The related LLM-generated explanations are depicted in Table 5.

3. Evaluation

Properties of LLM-based explanations. In Table 6, we summarize the different argumentation lines generated by the large language model (LLM). (1) In the context of the persuasion dimension *reciprocity*, LLM-generated explanations refer to the aspect of „giving something back to the community“, for example, purchasing an eco-friendly vehicle can be a way of giving back to the environment. (2) Explanations related to the persuasion dimension *scarcity* on the one hand refer to decreasing incentives for sustainable equipment (e.g., cars), on the other hand to limited resources (e.g., financial resources). (3) In the context of the persuasion dimension *authority*, different experts such as environmental experts are used as representatives of authorities. (4) Explanations related to the persuasion dimension *commitment* assume an existing commitment of the current user, for example, already proved in previous configuration sessions. (5) In the context of the persuasion dimension *liking*, LLM-generated explanations refer to a user's family, friends, and neighbors (e.g., your family will like your decision). (6) Explanations related to the persuasion dimension *social proof* refer to trends of peers, the wider community, and similar families.

LLM-based impact estimates. Using the LLM prompt *which of these explanations would have the highest persuasive impact on a user? Please provide a ranking.*, we asked the

Table 2
Sustainability-aware explanations related to the aspect of *purchasing a smaller car compared to an SUV*.

principle	LLM explanation
reciprocity	Consider the environmental impact and how it affects everyone. Choosing a more eco-friendly vehicle can be your way of giving back to the community and environment. Others will likely appreciate and reciprocate your thoughtful decision, fostering a positive cycle of environmentally conscious choices.
commitment and consistency	You've always been committed to sustainability and reducing your carbon footprint. Opting for an SUV contradicts these values due to its higher emissions and fuel consumption. Staying consistent with your commitment to the environment means choosing a more fuel-efficient and eco-friendly car.
social proof	Look at the growing trend among your peers and the wider community towards hybrid and electric vehicles. Many people are switching to these options because they are better for the environment and more economical in the long run. Following this trend shows you're in tune with progressive, responsible choices.
authority	Environmental experts and automotive industry leaders are advocating for smaller, more efficient vehicles to combat climate change and reduce pollution. Following the guidance of these authorities can help you make a well-informed decision that aligns with expert recommendations.
liking	Your friends and family who care about the environment will appreciate and support your choice of a more sustainable vehicle. This decision can strengthen your relationships, as they see you aligning your actions with shared values and concerns.
scarcity	With increasing regulations and shifts towards renewable energy, fuel-efficient and eco-friendly vehicles may soon become the norm, potentially leading to fewer incentives for traditional gas-guzzling SUVs. Take advantage of the current incentives for hybrid or electric vehicles before they become more scarce.

Table 3
Sustainability-aware explanations related to the aspect of *purchasing a standard compared to a long-range battery*.

principle	LLM explanation
reciprocity	Think about the balance between what you get and what you give. By opting for a standard-range battery, you're reducing the environmental impact associated with producing larger batteries. This considerate choice benefits the community and environment, and others may appreciate your mindful decision, leading them to reciprocate with similar sustainable choices.
commitment and consistency	You've been committed to reducing your carbon footprint and promoting sustainability. A standard-range battery is more aligned with these values, as it requires fewer resources to produce and has a lower environmental impact. Staying consistent with your commitment to sustainability means choosing the option that minimizes ecological harm.
social proof	Many environmentally conscious drivers are opting for electric vehicles with standard-range batteries because they strike a good balance between range and environmental impact. By choosing a standard-range battery, you're aligning with a growing community of eco-friendly drivers who prioritize sustainability over convenience.
authority	Experts in environmental science and sustainable technology often highlight the advantages of smaller, more efficient batteries. They point out that while long-range batteries offer more convenience, the production and disposal of larger batteries have a higher environmental cost. Following the advice of these authorities can help you make a more environmentally responsible decision.
liking	Your friends and family who are also concerned about environmental issues will likely appreciate your choice of a standard-range battery. This decision can enhance your relationships as they see you making sacrifices for the greater good and staying true to your shared values.
scarcity	As technology advances, the efficiency of electric vehicle batteries is continually improving. Standard-range batteries are becoming more capable, offering longer ranges and better performance. By choosing a standard-range battery now, you can take advantage of current technologies and innovations without feeling the pressure to invest in a long-range battery that may soon become less necessary.

LLM also for a ranking of the impact of the generated explanations following the idea of LLM self-evaluation [22]. In most settings, the dimensions *authority* and *commitment & consistency* have been regarded as the two most impactful persuasion dimensions (see Table 7). The related LLM-based argument is that the latter appeals directly to a person's desire to act in accordance with their past beliefs and statements and the former has a high impact specifically in scenarios where the underlying topic involves technical details. Finally, the dimension of *social proof* can be impactful in situations where social trends and peer behaviors impact decisions.

Feedback of study participants. For a very first evaluation beyond the LLM feedback, we conducted a small user study in our research group with N=10 participants (computer science students at our university). The participants were asked to rank the different explanations according to their potential persuasive impact. The results are included in Table 7. Overall, there exists an overlap between the explanations ranked highest by the LLM and those selected by the study participants. In this context, the *authority* principle has been regarded as relevant in all example configuration scenarios. This initial result indicates basic LLM capabilities to recommend persuasion strategies.

4. Threats to Validity

The explanation concepts presented in this paper are based on one selected configuration scenario (car configuration). These explanations are just high-level examples and many further (also more detailed ones) can be envisioned for car configuration (e.g., the sustainability aspects of a less powerful car engine) and beyond. In our work, we did not focus on a specific phase of a configuration process, i.e., the mentioned explanations could even be used before the configuration process has been started (e.g., as explanations in wish lists or product information). Gaining more related insights is a major focus of our future research.

Table 4
Sustainability-aware explanations related to the aspect of *waiving a car in the city center.*

principle	LLM explanation
reciprocity	Consider how much the city has invested in developing a robust public transportation system. By using these resources instead of buying a car, you show appreciation and reciprocate the city's efforts to provide efficient and sustainable transportation options. This can inspire others to also rely on public transport, enhancing the community's collective effort to reduce traffic and pollution.
commitment and consistency	You've always been committed to living a minimalist and environmentally friendly lifestyle. Owning a car might contradict these values, adding unnecessary expenses and contributing to urban pollution. Staying consistent with your principles means continuing to use the excellent public transportation available to you.
social proof	Many city dwellers are moving away from car ownership in favor of public transportation, biking, and walking. This trend reflects a growing recognition of the benefits of car-free living in urban areas. By following this trend, you align yourself with a progressive community that prioritizes sustainability and efficient urban living.
authority	Urban planning experts and environmental authorities advocate for reduced car usage in city centers to decrease congestion and pollution. By heeding their advice and utilizing public transport, you're supporting proven strategies that improve city life and contribute to a healthier environment.
liking	Your friends and neighbors who also rely on public transportation will likely appreciate and support your decision to avoid car ownership. This shared experience can strengthen your bonds with those around you, as you collectively benefit from less traffic, cleaner air, and a more vibrant urban community.
scarcity	Parking spaces and road capacity in the city center are limited resources. By not purchasing a car, you're helping to reduce the demand for these scarce resources, making the urban environment more livable for everyone. Additionally, public transport allows you to avoid the hassle and expense associated with finding and paying for parking.

Table 5
Sustainability-aware explanations related to the aspect of *purchasing an affordable car.*

principle	LLM explanation
reciprocity	Think about the support you've received from friends, family, and community during tough financial times. By opting for a more affordable car, you demonstrate financial responsibility and can reciprocate their support by being in a better position to help others when they need it.
commitment and consistency	You've always been committed to providing the best for your family while managing your limited financial resources wisely. Purchasing an expensive car contradicts this commitment and could put unnecessary financial strain on your household. Staying consistent with your goal of financial stability means choosing a car that fits within your budget.
social proof	Many families in similar financial situations prioritize practicality and affordability when it comes to transportation. By choosing a more economical vehicle, you're following a proven path that many others have taken to ensure they can meet their family's needs without financial stress.
authority	Financial advisors and experts consistently recommend living within your means and avoiding large, unnecessary expenses, especially when money is tight. By following this expert advice, you're making a prudent decision that aligns with sound financial practices and secures your family's financial future.
liking	Your friends and family will likely admire and respect your decision to prioritize your family's well-being over the allure of an expensive car. Making a practical choice shows that you value their advice and shared values of financial prudence and responsibility.
scarcity	Financial resources are currently scarce for your family. By not purchasing an expensive car, you conserve these limited resources for more critical needs like education, healthcare, and everyday living expenses. This ensures that you have the financial flexibility to handle unexpected costs and opportunities that arise.

The presented impact ranking of explanations has been primarily discussed on the basis of an LLM-generated ranking [22] including corresponding argumentations that help to understand the proposed ranking. More detailed studies with real users (and more detailed related preference and context information) are planned within the scope of future work also to better understand the limitations of LLMs with regard to the recommendation of persuasion strategies. Up to now, no LLM-related hallucination effects could be observed, however, this is an important aspect to be taken into account in future work. A recently mentioned new persuasion principle (identification) [23] will be taken into account in future studies. Finally, more detailed LLM prompts better taking into account the context (and preferences) of the current user are regarded as an important topic of future work.

5. Conclusions

In this paper, we have introduced the concept of sustainability-aware explanations of configurations. Using the example of car configuration, we have explained and exemplified this type of explanation. Following a set of persuasion dimensions, we have analyzed the LLM-generated explanations with regard to the used argumentation lines and analyzed the impact of the generated explanations on the user. In this context, LLMs

Table 6
LLM-based argumentation lines for Cialdini's principles of persuasion[20].

principle	argumentation line
reciprocity	giving something back to the community and the environment
scarcity	fewer incentives for sustainable equipment, limited available financial resources
authority	environmental experts, urban planning experts, financial advisors
commitment and consistency	existing commitment to sustainability in the past
liking	family, friends, neighbors
social proof	trends of peers, wider community, and similar families

Table 7
Scenario-dependent preferred explanations (top-2 LLM and study participant-preferred explanations).

scenario	top-2 (LLM)	top-2 (study participants)
SUV vs. smaller car	(1) commitment & consistency, (2) authority	(1) authority, (2) liking
Long vs. standard range battery	(1) authority, (2) commitment & consistency	(1) authority, (2) social proof
Car not needed in city center	(1) social proof, (2) authority	(1) authority, (2) reciprocity
Less costly car due to financial situation	(1) authority, (2) commitment & consistency	(1) authority, (2) commitment & consistency

show to be applicable in terms of generating explanations in a flexible fashion but also to recommend explanations in specific configuration contexts. Our future work will include detailed studies with real users with the goal to compare LLM-based rankings with the perception of explanations by real users. Further research will include an analysis of the effects of combining explanations (e.g., integrating authority-based with commitment-based explanations).

References

[1] A. Felfernig, M. Wundara, T. Tran, S. Polat-Erdeniz, S. Lubos, M. E. Mansi, D. Garber, V. Le, Recommender systems for sustainability: overview and research issues, Frontiers in Big Data 6 (2023). URL: https://www.frontiersin.org/articles/10.3389/fdata.2023.1284511. doi:10.3389/fdata.2023.1284511.

[2] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann, 2014.

[3] D. Sabin, R. Weigel, Product configuration frameworks-a survey, IEEE Intelligent Systems 13 (1998) 42–49. doi:10.1109/5254.708432.

[4] M. Stumptner, An Overview of Knowledge-based Configuration, AI Communications 10 (1997) 111–126.

[5] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-Driven Design, Analysis and Applications, SpringerBriefs in Computer Science, Springer, Cham, 2024. doi:10.1007/978-3-031-61874-1.

[6] B. J. Pine, B. Victor, A. Boynton, Making mass customization work, Harvard Business Review 71 (1993) 108–119.

[7] N. Tchertchian, D. Millet, Optimization approach for attractive and sustainable products, Procedia CIRP 90 (2020) 350–354. doi:10.1016/j.procir.2020.01.103.

[8] R. Wiezorek, N. Christensen, Integrating sustainability information in configurators, in: Configuration Workshop (ConfWS), 2021, pp. 58–64. URL: https://ceur-ws.org/Vol-2945/52-RW-ConfWS21_paper_16.pdf.

[9] T. N. T. Tran, A. Felfernig, V. M. Le, T. M. N. Chau, T. G. Mai, User needs for explanations of recommendations: In-depth analyses of the role of item domain and personal characteristics, in: 31st ACM Conference on User Modeling, Adaptation and Personalization, UMAP '23, ACM, New York, NY, USA, 2023, pp. 54–65. URL: https://doi.org/10.1145/3565472.3592950. doi:10.1145/3565472.3592950.

[10] G. Friedrich, Elimination of spurious explanations, in: 16th European Conference on Artificial Intelligence, ECAI'04, IOS Press, NLD, 2004, pp. 813–817.

[11] G. Friedrich, M. Zanker, A taxonomy for generating explanations in recommender systems, AI Magazine 32 (2011) 90–98. doi:10.1609/aimag.v32i3.2365.

[12] S. P. Erdeniz, M. Schrempf, D. Kramer, P. P. Rainer, A. Felfernig, T. Tran, T. Burgstaller, S. Lubos, Computational evaluation of model-agnostic explainable ai using local feature importance in healthcare, in: J. M. Juarez, M. Marcos, G. Stiglic, A. Tucker (Eds.), Artificial Intelligence in Medicine, Springer Nature Switzerland, Cham, 2023, pp. 114–119.

[13] A. Felfernig, M. Schubert, S. Reiterer, Personalized diagnosis for over-constrained problems, in: 23rd International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 1990–1996.

[14] V.-M. Le, A. Felfernig, T. N. T. Tran, M. Uta, Informedqx: Informed conflict detection for over-constrained problems, AAAI Conference on Artificial Intelligence 38 (2024) 10616–10623. doi:10.1609/aaai.v38i9.28932.

[15] B. O'Sullivan, A. Papadopoulos, B. Faltings, P. Pu, Representative explanations for over-constrained problems, in: 22nd National Conference on Artificial Intelligence - Volume 1, AAAI'07, AAAI Press, 2007, pp. 323–328.

[16] N. Tintarev, J. Masthoff, Evaluating the effectiveness of explanations for recommender systems, User Modeling and User-Adapted Interaction 22 (2012) 399–439. doi:10.1007/s11257-011-9117-5.

[17] N. Tintarev, Explanations of recommendations, in: ACM Conference on Recommender Systems, RecSys '07, Association for Computing Machinery, New York, NY, USA, 2007, pp. 203–206. doi:10.1145/1297231.1297275.

[18] N. Tintarev, J. Masthoff, A survey of explanations in recommender systems, in: ICDEW '07, IEEE Computer Society, USA, 2007, pp. 801–810. doi:10.1109/ICDEW.2007.4401070.

[19] A. Felfernig, B. Gula, G. Leitner, M. Maier, R. Melcher,

E. Teppan, Persuasion in knowledge-based recommendation, in: H. Oinas-Kukkonen, P. Hasle, M. Harjumaa, K. Segerståhl, P. Øhrstrøm (Eds.), Persuasive Technology, Springer, Berlin, Heidelberg, 2008, pp. 71–82.

[20] R. Cialdini, Influence: The Psychology of Persuasion, Quill, New York, NY, USA, 1993.

[21] S. Lubos, A. Felfernig, T. Tran, D. Garber, M. Mansi, S. Erdeniz, V. Le, Leveraging llms for the quality assurance of software requirements, in: Proceedings of the 32nd IEEE International Requirements Engineering 2024 Conference (RE@Next! Track), IEEE, 2024, p. to appear.

[22] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, X. Xie, A survey on evaluation of large language models, ACM Trans. Intell. Syst. Technol. 15 (2024). doi:10.1145/3641289.

[23] R. Cialdini, Influence, New and Expanded: The Psychology of Persuasion, HarperCollins, 2021.

Semantics-Preserving Merging of Feature Models

Mathias Uta[1], Viet-Man Le[2], Alexander Felfernig[2], Damian Garber[2], Gottfried Schenner[3] and Thi Ngoc Trang Tran[2]

[1]Siemens Energy AG, Erlangen, Germany
[2]Graz University of Technology, Graz, Austria
[3]Siemens AG, Vienna, Austria

Abstract
Large and globally operating enterprises can be confronted with situations where local variability models representing the constraints of individual countries and markets have to be integrated to support a centralized variability management. For example, a car producer operating in the U.S. as well as the European market, could be interested in having a centralized variability (feature) model representing the variability spaces of all supported markets. To achieve this goal, existing local feature models and the corresponding knowledge bases have to be integrated in such a way that the configuration spaces remain the same, for example, for the European market, we would request to support exactly the same set of car configurations that are supported by the corresponding local feature model. In this paper, we introduce an algorithmic approach that supports the merging of feature models in such a way that the semantics of the original feature models is preserved. We present our algorithm and the results of a solver performance analysis which has been conducted on the basis of real-world feature models.

Keywords
Variability Modeling, Feature Models, Model Merging, Redundancy Elimination, Configuration

1. Introduction

Feature models (FMs) are an intuitive way of representing commonality and variability properties of complex systems [1, 2, 3]. Specifically, in scenarios where companies are operating on a global basis, integration scenarios can arise where country or region-specific feature models have to be integrated to support a more globalized variability management. Think about a scenario where a car producer operating in the European and the US market decides to centralize variability management activities. On the technical (feature model) level, formerly region- or country-specific models have to be integrated in a systematic fashion in one centralized variability model. In this paper, we present an algorithmic approach to integrate (merge) two different ("old") feature models (e.g., the feature model $FMUS_{old}$ could denote a local feature model of a US car provider) in a semantics-preserving way where the solution (configuration) spaces of the local feature models are "transferred" to an integrated feature model which reflects exactly the same set of solutions: $solutions(FMUS_{old}) \cup solutions(FMEU_{old})$ equals $solutions(FM_{new})$. In this context, we assume that $FMUS_{old}$ and $FMEU_{old}$ represent the local feature models of a globally operating car manufacturer and FM_{new} is the result of merging the local feature models (and related knowledge bases).

Knowledge base merging has been approached in various ways. For example, the *alignment* of knowledge bases is based on the idea of knowledge base integration by identifying concepts in different knowledge bases that represent the same underlying concept but are represented by different names. Knowledge base alignment is specifically performed in situations where numerous knowledge bases have to be integrated [4]. Knowledge base *merging* is based on a set of predefined merging operations [5, 6], for example, consistency-based merging follows the goal of deriving a maximally consistent set of logical formulae that represent the union of the formulae of the original knowledge bases. Such integrations basically follow the idea of generating maximally satisfiable subsets (of rules) [7], i.e., sets that cannot be further extended (with original rules) without making the resulting knowledge base inconsistent.

Feature model merging [8, 9, 10, 11, 12, 13] is also in the line of the ideas of the previously mentioned approaches. Feature models can become quite large and complex [14], which makes the development and maintenance of single models a challenging task. Following the idea of *separation of concerns* [15], Aydin et al. [16] propose an approach to construct stakeholder-individual feature models which are then merged for the purpose of providing a unified view on the feature space. In the context

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain
mathias.uta@siemens-energy.com (M. Uta); vietman.le@ist.tugraz.at (V. Le); alexander.felfernig@ist.tugraz.at (A. Felfernig); dgarber@ist.tugraz.at (D. Garber); gottfried.schenner@siemens.com (G. Schenner); ttrang@ist.tugraz.at (T. N. T. Tran)
0000-0002-1670-7508 (M. Uta); 0000-0001-5778-975X (V. Le); 0000-0003-0108-3146 (A. Felfernig); 0000-0002-3550-8352 (T. N. T. Tran)

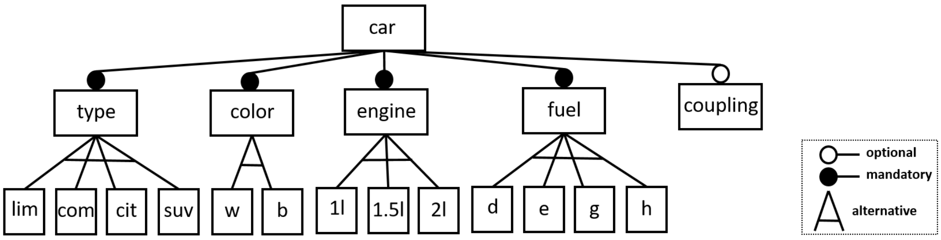


Figure 1: Example basic feature model from the automotive domain where *type* refers to the car type which can be (lim)ousine, (com)bi, (cit)y, and suv. Furthermore, the car *color* can be (b)lack or (w)hite, the *engine* can be 1l, 1.5l, and 2l. *Fuel* can be (d)iesel, (e)lectric, (g)asoline, and (h)ybrid, representing the supported types of fuel. Finally, a *coupling* unit is regarded as an optional feature.

of such a merging process, different "issues" have to be resolved, for example, some stakeholders regard a feature as optional while others think it should be mandatory. Furthermore, depending on the given scenario, feature naming can also become an issue if no "maximum feature set" has been specified ahead of the merging process. For such scenarios, Aydin et al. [16] propose a standard merging procedure that is able to generate a reference feature model, which then serves as a basis for further discussions and decision-making.

With a similar motivation, i.e., making large feature model development easier, Acher et al. [8], propose a set of integration operations for "local" feature models which basically support the goal of integrating local models into a global one. In this context, the authors also specify feature model relationships on a logical basis, for example, one feature model $FM1$ is the specialization of a feature model $FM2$ if the configuration space of $FM1$ is a subset of the configuration space of $FM2$ – see also Thüm et al. [17, 3]. The authors also introduce a *merge* operation where the introduced semantics does not support semantics preservation but requires that the result of the merging operation is equivalent or a superset of the solution (configuration) spaces of the two original feature models, i.e., $solutions(FM1) \cup solutions(FM2) \subseteq solutions(merge(FM1, FM2))$. Such a semantics of a merge operation is also considered in the contributions of Broek et al. [10], Carbonell et al. [11], and She et al. [18].

Following the *union* merge semantics introduced in Schobbens et al. [12], the feature model merging approach presented in this paper focuses on the preservation of the semantics of the source feature models used as an input for the merging procedure. In other words, it supports a semantics-preserving merging where the configuration space of the feature model resulting from a merging operation is exactly the union of the configuration spaces of the original feature models: $solutions(FM1) \cup solutions(FM2) = solutions(merge(FM1, FM2))$ which is more restrictive compared to the union semantics introduced by Acher et al. [8].

Compared to related work on feature model semantics preservation [10, 13], our approach provides a generalization in terms of (1) supporting arbitrary constraint types (in contrast to specific feature model related constraints such as *requires* and *incompatible*) and (2) taking into account redundancy-freeness in terms of assuring that redundant constraints as a result of a merging procedure can be detected and eliminated from the feature model. In our approach, the original feature models and the resulting feature model (result of the merging operation) are represented as constraint satisfaction problems (CSPs) [19]. To demonstrate the applicability of our approach, we present the results of a corresponding performance analysis.

The remainder of this paper is structured as follows. In Section 2, we introduce a working example consisting of simplified feature models from the automotive domain. Using this example, we discuss our algorithmic approach to semantics-preserving feature model merging in Section 3. To show the performance of our approach, we report the results of a corresponding performance evaluation (see Section 4). Finally, we conclude the paper with a discussion of existing threats to validity (Section 5) and a corresponding summary of the contributions of this paper (Section 6).

2. Example Scenario

We now introduce a simplified example of a feature model merging scenario. Our basic underlying assumption is that the original feature models are consistent, i.e., it is possible that at least one solution can be identified and also that the feature set of the original models are the same, i.e., the differences are primarily observable in terms of the constraints defined in the individual models. In our example from the automotive domain, the original feature models are denoted as $FMUS_{old}$ (the original US feature model) and $FMEU_{old}$ which denotes the original European Union feature model. In this context, we assume that these feature models are consistent, i.e., non-void [20], meaning that at least one configuration can be identified for each of those models. Finally, we denote the resulting model (the merging result) as FM_{new}.

Figure 1 represents the basic feature model (i.e., configuration model [21]) that in the following will be used as a working example. This feature model represents all relevant features that can be used to define variability knowledge, i.e., we assume that the same set of features is used to represent variability knowledge in $FMEU_{old}$ and $FMUS_{old}$. Differences in the two variability models can exist in terms of constraints representing individual configuration spaces. In the following, we specify constraints that define the properties of the two original feature models $FMEU_{old}$ and $FMUS_{old}$ represented in terms of individual constraint satisfaction problems (CSPs) representing the European and the US feature model [19].[1] These CSPs are defined in terms of variables with corresponding domain definitions (e.g., *type(lim,com,sit,suv)* denotes the variable *type* with the allowed values) and a corresponding set of constraints [22].

Note that *region* is an additional variable representing a contextual information, i.e., to which region a generated configuration belongs to. Contexts follow the idea of *separation of concerns* [15] which supports a kind of decentralized modeling [23]. For example, using the context variable *region*, the constraint $c_{1us} : fuel \neq h$ would be expressed as $c_{1us} : region = US \rightarrow fuel \neq h$ explicitly indicating that this constraint has to hold for configurations generated on the basis of the $FMUS_{old}$ CSP.

- $FMUS_{old}$: {*region(US), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, $c_{1us} : fuel \neq h$, $c_{2us} : fuel = e \rightarrow coupling = no$, $c_{3us} : fuel = d \rightarrow color = b$}
- $FMEU_{old}$: {*region(EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, $c_{1eu} : fuel \neq g$, $c_{2eu} : fuel = e \rightarrow coupling = no$, $c_{3eu} : fuel = d \rightarrow type \neq cit$}

To show the differences between the feature models $FMEU_{old}$ and $FMUS_{old}$, Table 1 provides an overview of the number of solutions supported by the original (region-specific) feature models.

3. Merging Feature Models

In order to be able to merge the two original feature models ($FMEU_{old}$ and $FMUS_{old}$) in a semantics-preserving fashion, each constraint of the two original feature models (represented as CSPs) has to be contextualized using the context variable *region*.[2] Assuming the two regions European Union and US, our context variable could be defined as *region(EU,US)* denoting the variable *region* with the allowed values $\{EU, US\}$. More precisely, each constraint $c_{[i]eu}$ ($c_{[i]us}$) of the "EU" ("US") CSP (derived from the $FMEU_{old}$ ($FMUS_{old}$) feature model) has to be translated into a contextualized representation – see the following example: $c_{1eu} : fuel \neq g$ would be translated into a corresponding contextualized form $c_{1eu'} : region = EU \rightarrow (fuel \neq g)$. The resulting contextualized variants of the original knowledge bases $FMEU_{old}$ and $FMUS_{old}$ are denoted as $FMEU'_{old}$ and $FMUS'_{old}$.

Table 1
Number of consistent solutions (configurations) related to the original and contextualized feature models.

Feature model	#configurations
$FMEU_{old}$	108
$FMUS_{old}$	96
$FM' = FMEU'_{old} \cup FMUS'_{old}$	204
$FMEU'_{old} \cap FMUS'_{old}$	84

- $FMUS'_{old}$: {*region(US), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, $c'_{1us} : region = US \rightarrow (fuel \neq h)$, $c'_{2us} : region = US \rightarrow (fuel = e \rightarrow coupling = no)$, $c'_{3us} : region = US \rightarrow (fuel = d \rightarrow color = b)$}
- $FMEU'_{old}$: {*region(EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no)*, $c'_{1eu} : region = EU \rightarrow (fuel \neq g)$, $c'_{2eu} : region = EU \rightarrow (fuel = e \rightarrow coupling = no)$, $c'_{3eu} : region = EU \rightarrow (fuel = d \rightarrow type \neq cit)$}

Note that the solution (configuration) spaces of the contextualized feature models $FMEU'_{old}$ and $FMUS'_{old}$ are the same as those of the original ones (assuming a corresponding context setting, e.g., *region = EU*). Following this argumentation, $solutions(FMEU_{old}) \cup solutions(FMUS_{old}) = solutions(FMEU'_{old} \cup FMUS'_{old})$ which supports our goal of achieving a semantics-preserving merging of the original knowledge bases (see Table 1).

The algorithmic approach to support such a semantics-preserving merging is shown in Algorithm 1 (MergeFM) which itself is a Flama [24] prototype implementation. In a first step (starting with line 6 of MergeFM), those constraints in the contextualized original knowledge bases

[1]The feature name abbreviations of $FMEU_{old}$ and $FMUS_{old}$ are defined in Figure 1.

[2]In general, contexts can be represented by a set of variables (i.e., not necessarily one).

(in Algorithm 1 denoted as FM_1' and FM_2') can be decontextualized where such a contextualization is not needed (c is a decontextualized version of c'): if $\neg c$ is consistent with $FM_1' \cup FM_2'$, there (obviously) exist solutions supporting $\neg c$. In such a case, the constraint c must be added in a contextualized fashion to the resulting knowledge base FM, since some feature model configuration (in the other knowledge base) supports $\neg c$. If $\neg c$ is inconsistent with $FM_1' \cup FM_2'$, c can be added in decontextualized fashion to the resulting knowledge base FM. In a second step (starting with line 14 of Algorithm 1), each constraint of the resulting knowledge base has to be checked for redundancy: in a logical sense, a constraint c can be regarded as redundant if $FM - \{c\}$ is inconsistent with $\neg c$ which means that the constraint does not reduce the solution space of FM and thus logically follows from the constraints in FM (and can be deleted from the constraints in FM).

Algorithm 1 MergeFM(FM_1', FM_2') : FM

1: $\{FM_1', FM_2'$: two contextualized and consistent feature models$\}$
2: $\{c'$: constraint c in contextualized form$\}$
3: $\{FM$: feature model as a result of MergeFM$\}$
4: $FM \leftarrow \{\}$;
5: $FM' \leftarrow FM_1' \cup FM_2'$;
6: **for all** $c' \in FM'$ **do**
7:     **if** *inconsistent*$(\{\neg c\} \cup FM' \cup FM)$ **then**
8:         $FM \leftarrow FM \cup \{c\}$;
9:     **else**
10:         $FM \leftarrow FM \cup \{c'\}$;
11:     **end if**
12:     $FM' \leftarrow FM' - \{c'\}$;
13: **end for**
14: **for all** $c \in FM$ **do**
15:     **if** *inconsistent*$((FM - \{c\}) \cup \{\neg c\})$ **then**
16:         $FM \leftarrow FM - \{c\}$;
17:     **end if**
18: **end for**
19: *return* FM;

When applying Algorithm 1 to $FMUS_{old}$ and $FMEU_{old}$, the resulting knowledge base FM_{new} looks like as follows. In the resulting knowledge base, the constraint c_{2us}' has been decontextualized. Also, as a result of applying Algorithm 1, constraint c_{2eu}' can be regarded as redundant and thus can be deleted from FM_{new}.[3]

- FM_{new}: {region(US,EU), type(lim,com,cit,suv), color(b,w), engine(1l, 1.5l, 2l), fuel(d, e, g, h), coupling(yes,no), $c_{1us}' : region = US \rightarrow (fuel \neq h)$, $c_{2us}' : fuel = e \rightarrow coupling = no$, $c_{3us}' : region = US \rightarrow (fuel = d \rightarrow color = b)$, $c_{1eu}' : region = EU \rightarrow (fuel \neq g)$, $c_{3eu}' : region = EU \rightarrow (fuel = d \rightarrow type \neq cit)$}

[3]Alternatively, c_{2us}' could be deleted as a redundant constraint (instead of c_{2eu}').

On the algorithmic level, the resulting knowledge base FM_{new} is represented in terms of a constraint satisfaction problem. One possibility of representing the integrated knowledge base as the resulting integrated feature model is depicted in Figure 2.

4. Performance Evaluation

In this section, we discuss the results of an initial performance analysis we have conducted to evaluate MergeFM (Algorithm 1) [4]. For this analysis, we applied 8 real-world variability models with varying sizes collected from the S.P.L.O.T. feature model repository [25] and the Diverso Lab's benchmark[5] [26]. Table 4 shows the characteristics of these models (denoted as ϕ). In order to generate "to-be-merged" feature models (FM_1' and FM_2') with different shares of contextualized constraints from individual ϕs, we determined the needed number of relationships or cross-tree constraints. We then modified these selected relationships/cross-tree constraints by changing their type, for example, changing mandatory to optional, changing alternative to or, or changing requires to excludes. The resulting models ($FM_1' \cup FM_2' = FM'$) are represented as constraint satisfaction problems [19] that differ individually in terms of the number of constraints (#constraints) and the degree of contextualization (expressed as percentages in Tables 2 and 3). In order to take into account deviations in time measurements, we repeated each experimental setting 10 times where in each repetition cycle the constraints in the individual (contextualized) knowledge bases FM' were ordered randomly. All analyses have been conducted with an Apple M1 Pro (8 cores) computer with 16-GB RAM. For evaluation purposes, we used the Choco solver[6] to perform the needed consistency checks.

The number of consistency checks needed for decontextualization is linear in terms of the number of constraints in FM'. A performance evaluation of MergeFM with different knowledge base sizes and degrees of contextualized constraints in FM is depicted in Table 2. In MergeFM, the runtime (measured in terms of milliseconds needed by the constraint solver[7] to find a solution) increases with the number of constraints in FM' and decreases with the number of contextualized constraints in

[4]The dataset, the implementation of algorithms, and evaluation programs can be found at *https://github.com/AIG-ist-tugraz/FMMerging*.
[5]https://github.com/flamapy/benchmarking
[6]choco-solver.org
[7]For the purposes of our evaluation we generated variability models represented as constraint satisfaction problems formulated using the Choco constraint solver – www.choco-solver.org.

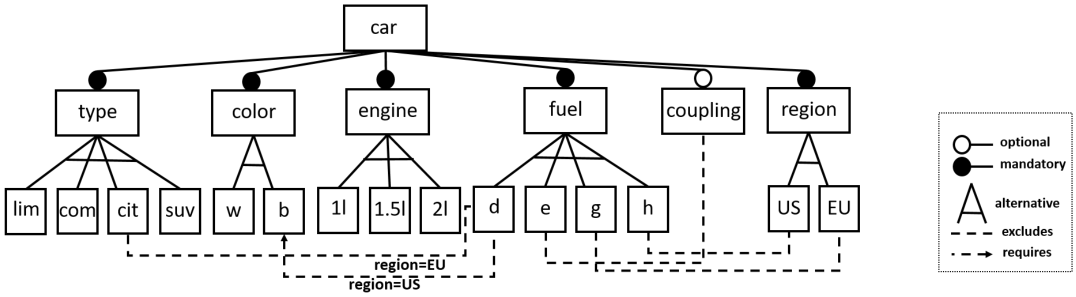


Figure 2: Example integrated feature model derived from FM_{new}. This model includes contextual information (the region) represented as feature(s). Simple contextualized constraints such as c'_{1us} : $region = US \rightarrow (fuel \neq h)$ are translated directly into a corresponding feature model constraint (as excludes relationship), for the representation of more complex constraints such as c'_{3eu} : $region = EU \rightarrow (fuel = d \rightarrow type \neq c)$, the corresponding feature model constraint is textually annotated with the context information (e.g., region=EU). This graphical representation of contexts in feature models follows the idea of contextual diagrams as introduced by Felfernig et. al [23].

Table 2
Avg. runtime (*in seconds*) of MergeFM measured with different *configuration knowledge base sizes* (of FM'_1 and FM'_2) and shares of related contextualized constraints (10-50% contextualization).

feature model (ϕ)	#constraints(ϕ)	10%	20%	30%	40%	50%
IDE	13	0.008	0.007	0.007	0.006	0.006
Arcade	66	0.060	0.056	0.054	0.052	0.054
FQA	101	2.560	2.341	2.794	2.812	3.684
Invest	233	3.018	3.860	4.879	5.781	5.915
Win8	405	154.825	171.516	165.988	158.998	149.323
EMB	1,029	1,621	1,361	1,138	1,043	972
EA	2,670	3,810	3,870	3,899	4,023	4,032
Linux	13,972	45,641	52,711	47,516	56,536	57,034

Table 3
Avg. runtime (*msec*) of the merged configuration knowledge bases (FM) to calculate a configuration measured with different *knowledge base sizes* (of FM) and shares of contextualized constraints in FM (10-50% contextualization).

feature model (ϕ)	#constraints(ϕ)	10%	20%	30%	40%	50%
IDE	13	0.050	0.042	0.039	0.037	0.037
Arcade	66	0.069	0.057	0.060	0.053	0.055
FQA	101	0.072	0.069	0.071	0.072	0.079
Invest	233	4.755	2.992	2.742	2.346	2.293
Win8	405	3.832	4.058	5.385	4.695	4.413
EMB	1,029	22.034	24.190	25.029	25.603	26.980
EA	2,670	40.501	41.227	43.741	45.311	51.483
Linux	13,972	143.698	199.822	143.756	159.515	112.986

Table 4
Feature models used for MergeFM evaluation (IDE=IDE product line, Arcade=Arcade Game PL, FQA=Feature model for Functional Quality Attributes, Invest=Feature model for Decision-making for Investments on Enterprise Information Systems, Win8=Accessibility options provided by Windows 8 OS, EMB=EMB Toolkit, EA=EA 2468, Linux=Linux kernel version 2.6.33.3).

feature model (ϕ)	IDE	Arcade	FQA	Invest	Win8	EMB	EA	Linux
#features	14	61	178	366	451	1,179	1,408	6,467
#hierarchical constraints	11	32	92	41	267	862	1,389	6,322
#cross-tree constraints	2	34	9	192	138	167	1,281	7,650
#CSP constraints	13	66	101	233	405	1,029	2,670	13,972

FM. The increase in efficiency can be explained by the fact that a higher degree of contextualization includes more situations where the inconsistency check in Line 7 (Algorithm 1) terminates earlier (a solution has been found) compared to situations where no solution could be found. In addition, Table 3 indicates that the performance of solution search does not differ depending on the degree of contextualization in the resulting knowledge base *FM*.

Consequently, integrating individual variability models can trigger the following improvements. (1) De-contextualization in *FM* can lead to less cognitive efforts when adapting / extending knowledge bases (due to a potentially lower number of constraints [27] and a lower degree of contextualization). (2) Reducing the overall number of constraints in *FM* can also improve runtime performance of the resulting integrated knowledge base.

5. Threats to Validity

We are aware that our evaluation is in fact based on real-world feature models, however, synthesized variants thereof have been used for MergeFM evaluation purposes. Furthermore, our approach is based on the assumption that the "to-be-merged" feature models have the same set of features, i.e., we assume feature equivalence. In this context, we assume that in real-world scenarios further streamlining tasks (e.g., feature name alignment) have to be completed before MergeFM can be activated. Our basic assumption behind redundancy elimination and de-contextualization in MergeFM is that the understandability and maintainability of feature models can be improved – although already confirmed by related work [27], further research is needed to better understand major impact factors that make feature models (and underlying knowledge bases) easier to understand and maintainable.

6. Conclusions and Future Work

In this paper, we have introduced an approach to the consistency-based merging of variability models represented as constraint satisfaction problems. The approach helps to build semantics-preserving feature models in the sense that the solution spaces of the resulting knowledge bases (result of the merging process) correspond to the union of the solution spaces of the original knowledge bases. Such an approach can be useful in the mentioned integration scenario but as well in situations where different parts (representing different contexts) of a knowledge are developed in a de-centralized fashion and integrated thereafter. Besides the preservation of the original semantics, our approach also helps to make the resulting knowledge base compact in the sense of deleting redundant constraints and not needed contextual information. The runtime performance of our approach is shown on the basis of a first performance analysis with real-world feature models. Future work will include the evaluation of our concepts with further knowledge bases and the development of alternative merging algorithms with the goal to further improve runtime performance. Furthermore, we will evaluate different alternative feature model representations that help to represent contextualized constraints – one such representation has been discussed in this paper.

References

[1] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented domain analysis feasibility study (foda), Technical Report, CMU/SEI-90-TR-021 (1990).

[2] K. Czarnecki, S. Helsen, U. Eisenecker, Formalizing cardinality-based feature models and their specialization, SoftwareProcess: Improvement and Practice 10 (2005) 7–29.

[3] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-Driven Design, Analysis and Applications, SpringerBriefs in Computer Science, Springer, Cham, 2024. doi:10.1007/978-3-031-61874-1.

[4] L. Galarraga, N. Preda, F. Suchanek, Mining rules to align knowledge bases, in: Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, San Francisco, CA, 2013, pp. 43–48.

[5] J. Delgrande, T. Schaub, A consistency-based framework for merging knowledge bases, Journal of Applied Logic 5 (2007) 459–477.

[6] P. Liberatore, M. Schaerf, Arbitraton (or how to merge knowledge bases), IEEE Transactions on Knowledge and Data Engineering 10 (1998) 76–90.

[7] R. Reiter, A theory of diagnosis from first principles, AI Journal 23 (1987) 57–95.

[8] M. Acher, P. Collet, P. Lahire, R. France, Composing feature models, in: M. van den Brand, D. Gašević, J. Gray (Eds.), Software Language Engineering, Springer, Berlin, Heidelberg, 2010, pp. 62–81.

[9] V. Bischoff, K. Farias, L. Gonçales, J. Victória Barbosa, Integration of feature models: A systematic mapping study, Information and Software Technology 105 (2019) 209–225. URL: https://www.sciencedirect.com/science/article/pii/S0950584916302178. doi:https://doi.org/10.1016/j.infsof.2018.08.016.

[10] P. van den Broek, I. Galvao, J. Noppen, Merging feature models, in: 15th International Software Product Line Conference, Jeju Island, South Korea, 2010, pp. 83–90.

[11] J. Carbonnel, M. Huchard, A. Miralles, C. Nebut, Feature model composition assisted by formal concept analysis, in: 12th International Conference on Evaluation of Novel Approaches to Software Engineering, 2017, pp. 27–37. doi:10.5220/0006276600270037.

[12] P. Y. Schobbens, P. Heymans, J. C. Trigaux, Feature Diagrams: A Survey and a Formal Semantics, in: 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, MN, USA, 2006, pp. 139–148. doi:10.1109/RE.2006.23.

[13] S. Segura, D. Benavides, A. Ruiz-Cortés, P. Trinidad, Automated Merging of Feature Models Using Graph Transformations, in: Generative and Transformational Techniques in Software Engineering II, volume 5235 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 139–148. doi:10.1109/10.1007/978-3-540-88643-3_15.

[14] M. Acher, H. Martin, L. Lesoil, A. Blouin, J. Jézéquel, D. Khelladi, E. Djamel, O. Barais, J. Pereira, Feature Subset Selection for Learning Huge Configuration Spaces: The Case of Linux Kernel Size, in: 26th ACM International Systems and Software Product Line Conference - Volume A, ACM, 2022, pp. 85–96. URL: https://doi.org/10.1145/3546932.3546997. doi:10.1145/3546932.3546997.

[15] B. Nuseibeh, J. Kramer, A. Finkelstein, Viewpoints: meaningful relationships are difficult!, in: 25th International Conference on Software Engineering, 2003, pp. 676–681. doi:10.1109/ICSE.2003.1201254.

[16] E. A. Aydin, H. Oguztuzun, A. H. Dogru, A. S. Karatas, Merging Multi-view Feature Models by Local Rules, in: 9th International Conference on Software Engineering Research, Management and Applications, Baltimore, MD, USA, 2011, pp. 140–147. doi:10.1109/SERA.2011.34.

[17] T. Thüm, D. Batory, C. Kästner, Reasoning about edits to feature models, in: 31st International Conference on Software Engineering, ICSE '09, IEEE Computer Society, USA, 2009, pp. 254–264. URL: https://doi.org/10.1109/ICSE.2009.5070526. doi:10.1109/ICSE.2009.5070526.

[18] S. She, U. Ryssel, N. Andersen, A. Wąsowski, K. Czarnecki, Efficient synthesis of feature models, Information and Software Technology 56 (2014) 1122–1143. URL: https://www.sciencedirect.com/science/article/pii/S0950584914000238. doi:https://doi.org/10.1016/j.infsof.2014.01.012.

[19] E. Tsang, Foundations of Constraint Satisfaction, Academic Press, London, 1993.

[20] D. Benavides, S. Segura, A. Ruiz-Cortes, Automated analysis of feature models 20 years later: A literature review, Information Systems 35 (2010) 615–636.

[21] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1st ed., Morgan Kaufmann Publishers, 2014.

[22] D. Benavides, P. Trinidad, A. Ruiz-Cortes, Using constraint programming to reason on feature models, in: 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), Taipei, Taiwan, 2005, pp. 677–682.

[23] A. Felfernig, D. Jannach, M. Zanker, Contextual diagrams as structuring mechanisms for designing configuration knowledge bases in uml, in: 3rd International Conference on the Unified Modeling Language (UML2000), volume 1939 of *Lecture Notes in Computer Science*, Springer, York, UK, 2000, pp. 240–254.

[24] J. Galindo, J. Horcas, A. Felfernig, D. Fernandez-Amoros, D. Benavides, Flama: A collaborative effort to build a new framework for the automated analysis of feature models, in: 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 16–19. URL: https://doi.org/10.1145/3579028.3609008. doi:10.1145/3579028.3609008.

[25] M. Mendonca, M. Branco, D. Cowan, S.P.L.O.T.: Software Product Lines Online Tools, in: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA '09, ACM, New York, NY, USA, 2009, pp. 761–762. doi:10.1145/1639950.1640002.

[26] R. Heradio, D. Fernandez-Amoros, J. A. Galindo, D. Benavides, D. Batory, Uniform and scalable sampling of highly configurable systems, Empirical Software Engineering 27 (2022) 44.

[27] A. Felfernig, M. Mandl, A. Pum, M. Schubert, Empirical knowledge engineering: cognitive aspects in the development of constraint-based recommenders, in: Proceedings of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 631–640.

An extensive comparison of preprocessing methods in the context of configuration space learning

Damian Garber[1,*], Alexander Felfernig[1], Viet-Man Le[1], Tamim Burgstaller[1] and Merfat El-Mansi[1]

[1]Graz University of Technology, Inffeldgasse 16b, Graz, Styria, Austria

Abstract

One of the core goals in the research field of configuration space learning is building precise predictive models that allow for reliably estimating the performance of a configuration without requiring costly tests. The models used for this purpose are usually machine learning-based. However, the models show significant deviations in their performance depending on the investigated Software Product Line (SPL), the applied data preprocessing, and the number of sample configurations collected. Thus, we investigate the impact of different preprocessing methods and their behavior when using different SPLs, machine learning models, and sample sizes. Performance comparisons on this scale are usually not conducted due to their prohibitively expensive execution time requirements, even for smaller SPLs. Thus, we used three fully enumerated spaces as our training data, which allows for more generalized results. Our results show that the average factors between the worst and best-performing preprocessing methods are 2.05 (BerkeleyDBC), 1.17 (7z), and 1.84 (VP9). Further, no single preprocessing method tested was able to outperform all others, nor was this the case within one specific SPL or model type. This underlines the importance of testing new approaches with multiple preprocessing methods.

Keywords

Configuration Space Learning, Machine Learning, Preprocessing

1. Introduction

The discovery of configurations that optimize the performance of any given Software Product Line (SPL) is one of the core goals of configuration space learning. The performance of a model can take many forms and rely heavily on the use case. For instance, one may optimize a SPL to perform a core task very efficiently or optimize for the size of the compiled SPL binary. This optimization usually takes place in steps. The first step is sampling configurations from the configuration space of the SPL and measuring the target property, which often entails compiling and running tests or benchmarks, a very time and resource-intensive undertaking. One can use these samples to train a prediction model, which is then used to find a configuration that optimizes the target property. In this paper, we focus on the creation and training of the prediction model. Many factors can impact the performance of a performance prediction model for SPLs, from the SPL itself to the sampling approach used to collect the training data. However, our focus lies on one of the factors often neglected in SPL performance prediction: Preprocessing. We will define any necessary terms used in this paper in Section 2. Preprocessing has proven itself in many other domains that employ machine learning-based prediction models. However, literature reviews such as Gong and Chen [1] show that less than half of the investigated studies within the field of configuration performance learning use preprocessing, further discussed in Section 3. Accordingly, we thus conduct an in-depth investigation on the influence of preprocessing on performance prediction models for SPLs. To this end, we measure the performance of 4 preprocessing methods in the context of 3 SPLs, 5 machine learning models, and 20 different sizes of training sets. We discuss the details of the experimental evaluation in Section 4, followed by a discussion of the results in Section 5.

2. Definitions

Software Product Line (SPL). SPLs, as a concept started to gain widespread popularity at the beginning of the 2000s [2]. Engström and Runeson [3] describe SPLs as the paradigm of forming derivate products from a set of generic components. A SPL has multiple features, each supporting an individual domain of values, which allows for the generation of diverse products using the same components.

Configuration. In the context of a SPL, a configuration defines for each feature the corresponding feature value. However, there may exist additional constraints within

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.

dgarber@ist.tugraz.at (D. Garber); alexander.felfernig@ist.tugraz.at (A. Felfernig); vietman.le@ist.tugraz.at (V. Le); tamim.burgstaller@ist.tugraz.at (T. Burgstaller); merfat.elmansi@un.org (M. El-Mansi)

0009-0005-0993-0911 (D. Garber); 0000-0003-0108-3146 (A. Felfernig); 0000-0001-5778-975X (V. Le); 0009-0007-4522-8497 (T. Burgstaller); 0009-0005-2695-4210 (M. El-Mansi)

the SPL. Thus, we speak of a valid configuration if none of the assigned values is inconsistent with any of the constraints.
Configuration Space. The configuration space of a SPL describes the space spanned by all valid configurations of the SPL. The size of configuration spaces commonly grows exponentially with the number of features, and we speak of colossal configuration spaces if its size is $\gg 10^{10}$ [4].

Name	Value
Vendor	Lenovo
Product	20N6001GGE
CPU	Intel Core i7-8665U (4x 1,90 GHz)
RAM	32GB (DDR4)
OS	Manjaro Linux
Kernel version	6.1.80.-1-MANJARO

Table 1
Specifications of the machine used in the experiments

3. Related Work

The three fully enumerated configuration spaces provided by Oh et al.[4] facilitate the comprehensive performance analysis and comparison we conducted. They use them to show that relatively simple approaches like uniform random sampling can outperform well-established tools like SPL Conquerer [5] to find near-optimal configurations for SPLs. We build on this idea and conduct a comparison of different preprocessing methods. The reasoning behind conducting this comparison is the alarming result of Gong and Chen [1]. They performed a literature review on deep configuration performance learning and reported that 44 out of 85 investigated studies used the data as it was without any preprocessing. This limited utilization implies a lack of awareness of the impact of preprocessing methods. This lack of awareness may then aggravate the difficulty of reproducing and validating the results of published works. Dacrema et al. [6], for example, investigated 18 new approaches published recently, of which they could only reproduce 7. Of the 7 reproduced approaches, they showed that they can outperform 6 by using relatively simple other approaches. The importance of preprocessing methods in many domains has long since been established. Wu et al.[7], for example, shows that preprocessing improves the performance of streamflow forecasts. Rasekhi et al. [8] report improvements in the prediction of epileptic seizures by using preprocessing.
We further include in our tests different sample sizes, which allows us to investigate the reaction of the preprocessing methods to changing sample sizes. Acher et al.[9] sampled and measured 95854 Linux configurations, a minute fraction of the configuration space of 2^{15000} ($2.818 * 10^{4515}$) configurations spaned by Linux. They reported to have needed 15000 hours of computation time to collect the samples. Guo et al. [10] uses tree-based models to predict configuration performances. Martin et al.[11] focus on using transfer learning across different versions of the Linux kernel to predict the performances of the versions. They mention preprocessing only for encoding configurations into formats compatible with their machine-learning approach.
The literature reviews of Gong and Chen [1] and Pereira et al. [12] named multiple data sampling approaches used in configuration performance learning. However, both identified random sampling as the most popular approach. Pereira et al. [13] conducted a dedicated study on sampling approaches for learning configuration spaces. They suggest using uniform random sampling as long as it is computationally feasible. Accordingly, we adapted it for our comparison.

4. Experimental Setup

This section will discuss the exact experimental setup for data collection and which machine learning models, preprocessing methods, and datasets we were using. All measurements were collected using the same machine with specifications as they are listed in Table 1. We use Mean Absolute Percentage Error (MAPE) to evaluate the model performances, which is one of the most commonly used metrics in literature [1] [12] [13]. The code was implemented in Python using the widely used scikit-learn[1] library [14]. We used Uniform Random Sampling (URS) to generate the training sets of different sizes for model learning. We can perform URS by selecting configurations randomly from the set of valid configurations. The size of the training sets range from 50 to 1000 in steps of 50. However, the tests for all models and preprocessing methods use, within the same iteration, the same training set of a specific size. After the performance of all models using the preprocessing applied to the training sets is measured, these measurements are repeated 15 times, each time with new training sets selected with URS. The average of the resulting MAPE values in the 15 iterations is the value we use when we discuss the results.

4.1. Datasets

For our comparison, we use a dataset of fully enumerated configuration spaces. Thus, the dataset includes all valid configurations for a given SPL. In addition, a value, like execution times of benchmarks or similar, representing the performance of each configuration was measured. We use three such datasets based on three configurable

[1]https://scikit-learn.org/stable/index.html

software projects: **BerkeleyDBC**[2], **7z**[3], and **VP9**[4]. The datasets are used in the work of Oh et al.[4], and they were made available in their resources[5]. Oh et al.[4] provided the following description of the three datasets.

BerkeleyDBC is an embedded database system with 9 variables and 2560 configurations. Benchmark response times were measured. We visualize the variable names and their domains in Table 2.

Name	Domain
DIAGNOSTIC	0 | 1
HAVE_STATISTICS	0 | 1
HAVE_REPLICATION	0 | 1
HAVE_CRYPTO	0 | 1
HAVE_SEQUENCE	0 | 1
HAVE_VERIFY	0 | 1
HAVE_HASH	0 | 1
CACHESIZE	CS16MB | CS32MB | CS64MB | CS512MB
PAGESIZE	PS1K | PS4K | PS8K | PS16K | PS32K

Table 2
BerkeleyDBC variable names and their respective domains

7z is a file archiver with 9 variables and 68640 configurations. Compression times were measured. We visualize the variable names and their domains in Table 3.

Name	Domain
root	0 | 1
CompressionMethod	LZMA | LZMA2 | PPMd | BZip2 | Deflate
x	x_0", "x_2 | x_4 | x_6 | x_8 | x_10
BlockSize	BlockSize_1", "BlockSize_2 | BlockSize_4 | BlockSize_8 | BlockSize_16 | BlockSize_32 | BlockSize_64 | BlockSize_128 | BlockSize_256 | BlockSize_512 | BlockSize_1024 | BlockSize_2048 | BlockSize_4096
Files	Files_0 | Files_10 | Files_20 | Files_30 | Files_40 | Files_50 | Files_60 | Files_70 | Files_80 | Files_90 | Files_100
tmOff	0 | 1
mtOff	0 | 1
HeaderCompressionOff	0 | 1
filterOff	0 | 1

Table 3
7z variable names and their respective domains

VP9 is a video encoder with 12 variables and 216000 configurations. Video encoding times were measured. We visualize the variable names and their domains in Table 4.

Although these three configuration spaces do not approach the sizes of colossal configuration spaces like Linux, which spans a configuration space with 2^{15000} configurations[9], they still have sizes where an enumeration is no longer an option, and thus fall in the purview of the research field of configuration space learning. Depending on the complexity of the tests and the underlying system, procuring very few samples may already be very costly. Acher et al.[9], for example, reported 15000 hours of computation to build and measure 95854 Linux configurations. We selected these datasets due to two main advantages. The first is avoiding the extreme computation times of collecting such data, and the second is that using them allows us to test multiple iterations of training sets of different sizes.

4.2. Models

We selected five different types of machine-learning models, each representing a different general approach to maximize the usefulness of our results. In our implementations, we used models from the scikit-learn[6] python library [14]. For the sake of reproducibility, we did not perform any parameter tuning on the models and used their respective default settings if not explicitly stated otherwise.

The first model is a Multi-Layer Perceptron (**MLP**) model, a feedforward neural network approach. We set the maximum of iterations to 1000 and activated early stopping for our tests. MLPs are, according to Gong and Chen [1], the most popular approach when conducting deep configuration performance learning. However, it is a very data-intensive approach that needs comparatively large training sets to perform well.

The second model is a K-Nearest Neighbors (**KNN**) model, a memory-based approach. The model finds the k-nearest neighbors to a configuration from the training set, in our case the default value 5. The KNN model predicts the performance of the configuration by calculating the average of the performances of the configuration's k nearest neighbors. In our case, the average was weighted by the distance between the neighbor and the configuration.

The third model is a Random Forest (**RF**), an ensemble method employing several decision trees generated using the training data to predict the performance of an un-

[2]https://www.oracle.com/database/technologies/related/berkeleydb.html
[3]https://www.7-zip.org/download.html
[4]https://www.webmproject.org/vp9/
[5]https://zenodo.org/records/7776627
[6]https://scikit-learn.org/stable/index.html

Name	Domain
root	0 | 1
lagInFrames	lagInFrames_0 | lagInFrames_8 | lagInFrames_16
endUsage	variableBitrate | constantBitrate | constrainedQuality
AdaptiveQuantizationMode	off | variance | complexity | cyclicRefresh
TileColumns	TileColumns_0 | TileColumns_3 | TileColumns_6
cpuUsed	cpuUsed_0 | cpuUsed_2 | cpuUsed_4 | cpuUsed_6 | cpuUsed_8
Threads	Threads_2 | Threads_4 | Threads_6 | Threads_8 | Threads_10
bitRate	bitRate_300 | bitRate_600 | bitRate_900 | bitRate_1200 | bitRate_1500
FrameBoost	0 | 1
lossless	0 | 1
AutoAltRef	0 | 1
Quality	good | realtime

Table 4
VP9 variable names and their respective domains

known data point. We use bagged trees, which means we train all underlying decision trees to solve the problem using all features. The final result is in the context of classification decided based on a majority vote. However, in our context of regression, we calculate the final result by taking the mean of all results produced by the decision trees.
The fourth model is a Support Vector Machine (**SVM**), a well-established model based on statistical learning frameworks. We use a radial basis function as our kernel type.
The final model is an ElasticNet (**EN**) model, a derivate of linear regression models. The model combines L1 and L2 priors as a regularizer.

4.3. Preprocessing

We used several preprocessing methods to test their impact on the different models and training sizes. For the sake of this comparison, we do not distinguish between actual preprocessing methods like Standardization and encodings such as the One Hot Encoding.
The first preprocessing method discussed we call default (**DEF**). It provides a baseline for mostly unaltered data and leaves numeric values untouched. The boolean values are, however, encoded with 0 and 1 for false and true, respectively. If all values of a domain can be converted into numbers this is done (e.g. CS32MB = 32, Table 2). If this is not possible, the string values are encoded using label encoding [15, 16], which assigns an increasing numeric value for each unique string in a domain. This format is the default state of the data. Thus, we apply all preprocessing methods mentioned hereafter to the data in this format.
The second preprocessing method is Min Max Scaling (**MMS**) [17, 18], which reduces the scale of a given feature to be between 0 and 1. We achieve this by applying Equation 1 on every feature of the configuration, where min and max are the minimum and maximum recorded numbers for this feature, respectively. When we apply this to the features encoded using label encoding, the result is a derivative of the former called scaled label encoding [19, 20].

$$f_i(x_i) = \frac{x_i - min_i}{max_i - min_i} \tag{1}$$

The third preprocessing method is Standardization (**STD**) [21, 22], which is achieved by calculating the mean and standard deviation of each feature and applying Equation 2

$$f_i(x_i) = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

This results in the mean of every feature in the training set being now 0 and the standard deviation being 1.
The final preprocessing method is One Hot Encoding (**OHE**) [23, 24], which changes the domain of all features to a boolean domain. We achieve this by increasing the dimensions of the data by an encoding of the domain. Thus, if, for example, feature f has the domain $0, 6, 12$, it would have been replaced with the features f_0, f_6, f_{12} each of the three resulting boolean features are mutually exclusive and encode one possible value assigned to feature f.

5. Results

In this section, we showcase the measurements collected as described in the experimental setup section and discuss them. To this end, we will discuss the results of each dataset separately and what observations we made.
Firstly, we start with a discussion of our smallest SPL, BerkeleyDBC. All performance results are visualized in Figure 1. In the results for MLP, we see that OHE is performing best among all preprocessing methods regardless of sample size. However, we can also see a shift in the performances of the preprocessing approaches. STD started as the worst-performing preprocessing method. Despite that, with increasing sample size, it outperformed MMS and DEF. Accordingly, the results of STD approached the results of the best performer OHE for the larger sample sizes. However, when looking at Figure 2 and Figure

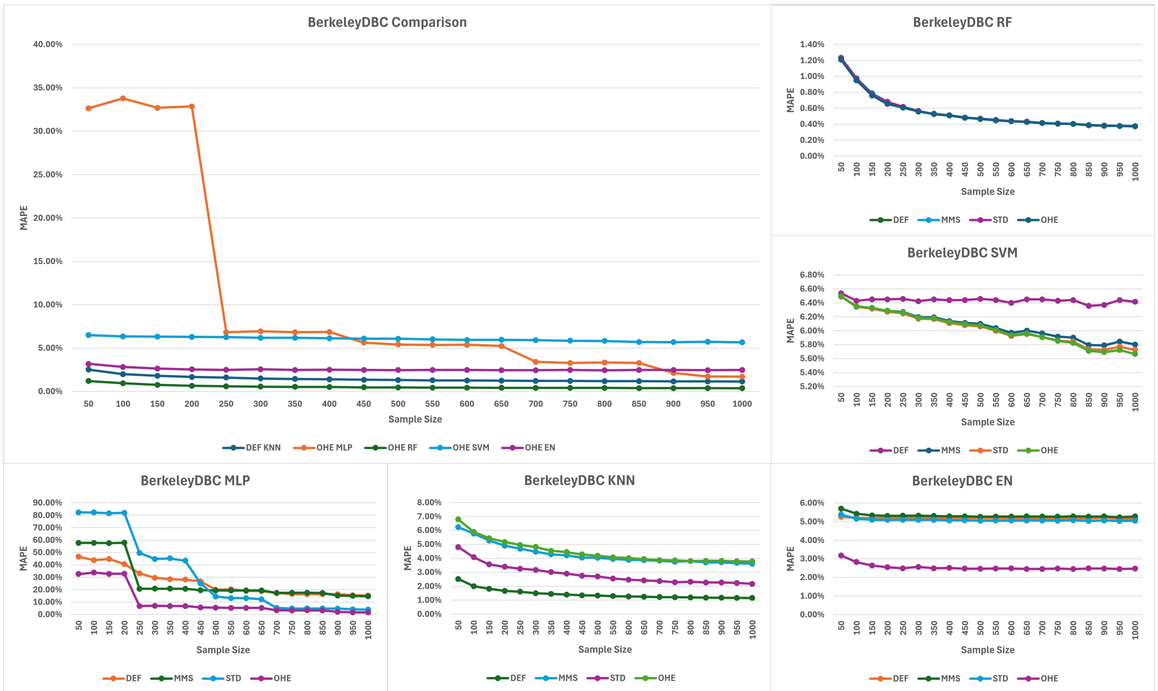


Figure 1: The performances of each preprocessing method applied to each model and a comparison between the top performers of each model aplyed to the BerkeleyDBC dataset.

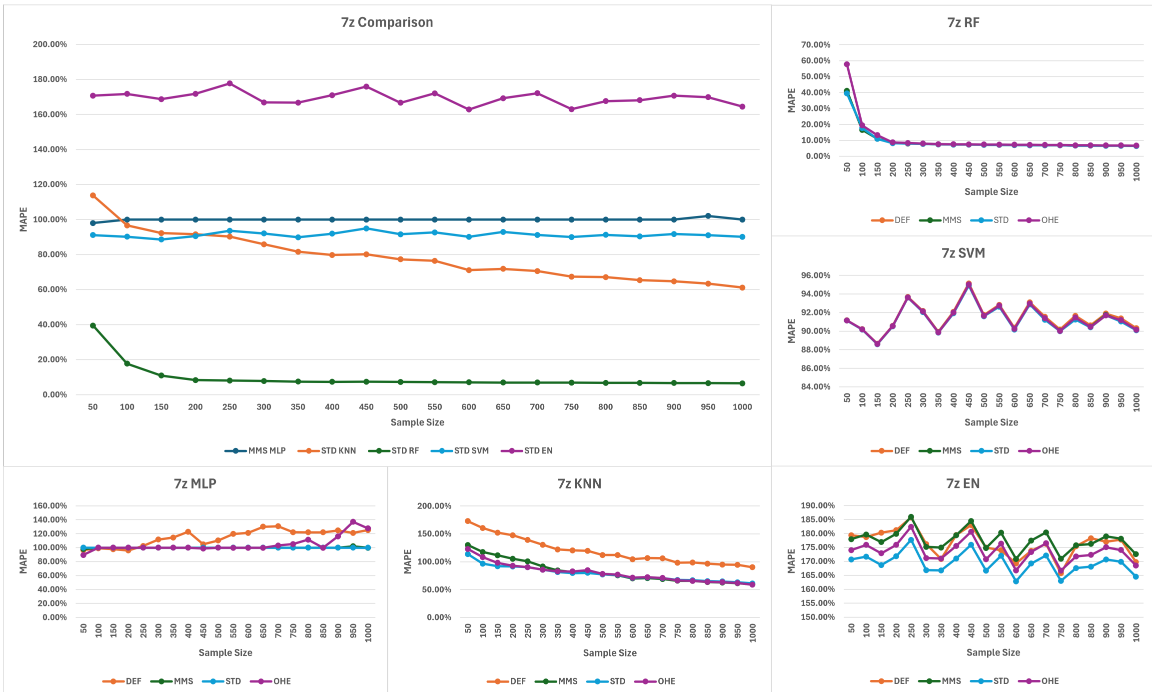


Figure 2: The performances of each preprocessing method applied to each model, and a comparison between the top performers of each model applied to the 7z dataset.

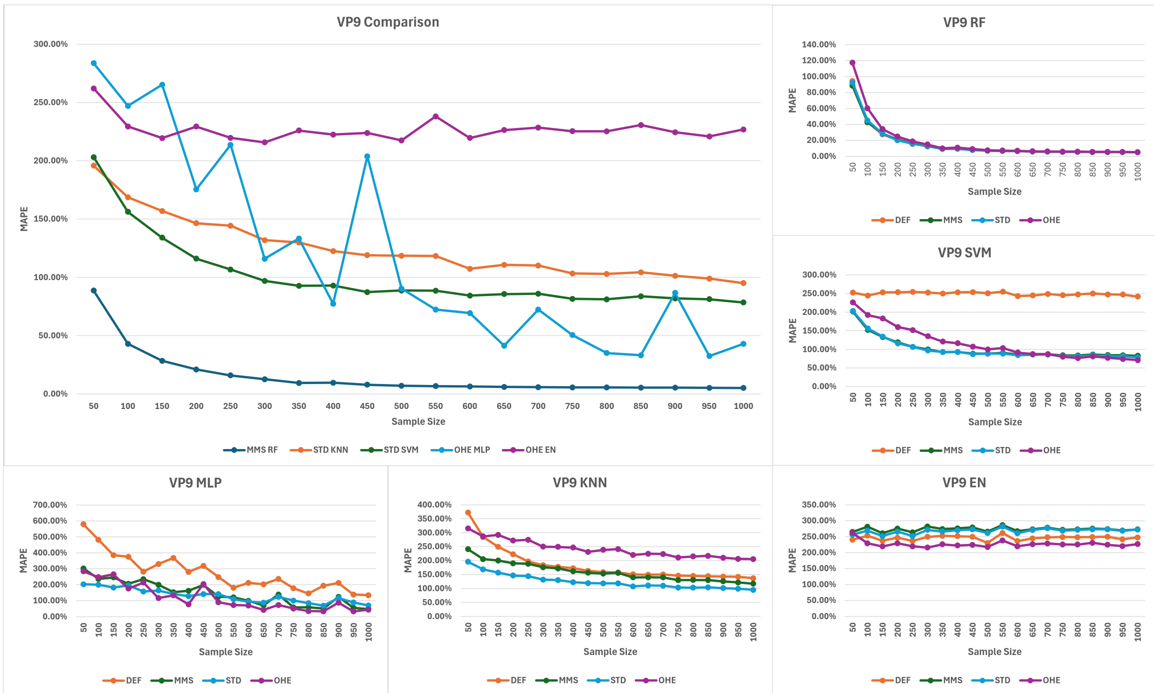


Figure 3: The performances of each preprocessing method applied to each model, and a comparison between the top performers of each model applied to the VP9 dataset.

3, we see that this behavior does not occur in the other datasets, but, in contrast to other approaches, the results of STD remain either relatively stable or improve with increasing sample sizes. We see a similar behavior on a smaller scale with MMS and DEF. MMS performed initially worse than DEF, overtaking it as soon as sample sizes became larger than 200 and achieving similar results from sample sizes 500 and larger. The other models, in contrast, showed more pronounced preferences for preprocessing methods. For the KNN model, DEF was the best-performing preprocessing method, followed by STD, MMS, and OHE. RF showed the best performance of all models with almost indistinguishable differences of 0.01% between the preprocessing methods on average. For the SVM model, DEF performed the worst with comparatively little improvement with larger sample sizes. The remaining preprocessing methods, from worst to best, MMS, STD, and OHE show relatively similar results, improving with increasing sample sizes. For the EN model, OHE performs best, and the remaining three, from worst to best, MMS, DEF, and STD show relatively similar results. The sample size has a comparatively small impact on the performances. Results with sample sizes of 200 and larger only show a minor oscillation and remain otherwise stable. The comparison between models shows that RF outperforms the other models. All models except for EN show significantly better performances with larger sample sizes. However, MLP was impacted the most by the sample size. It overtook the performance of SVM at a sample size of 450 and EN at 900.

Secondly, we will discuss the next larger SPL, 7z. We provide the results for this dataset in Figure 2. The first observation we can make is that the overall quality of the predicted results decreased. This matches our expectations, since we are predicting the performance of a larger SPL using an equivalent setup. Another observation we can make is that three of the five tested models showed strong oscillations in their performances or, for some preprocessing methods, a worsening of the performance with increasing sample size. The MLP model, for example, showed for the best and second best performing preprocessing methods, MMS and STD, respectively, no significant changes with increasing sample sizes. DEF and OHE showed meanwhile a decrease in performance with increasing sample sizes. The EN and SVM models showed strong oscillations with increasing sample sizes. However, the performances of the preprocessing methods all follow that same pattern, which suggests that the cause for this may lie in the model or the SPL rather than the preprocessing approaches. In the case of the SVM, the performances remained very similar. The preprocessing performances in the EN model follow the same oscillation

Model	Preprocessing	BerkeleyDBC	7z	VP9
MLP	DEF	25.68%	114.76%	273.94%
MLP	MMS	26.29%	**99.97**%	144.24%
MLP	STD	31.12%	99.98%	129.80%
MLP	OHE	**10.27**%	104.41%	**117.11**%
KNN	DEF	**1.44**%	119.01%	182.27%
KNN	MMS	4.29%	82.30%	158.79%
KNN	STD	2.85%	**78.42**%	**124.31**%
KNN	OHE	4.47%	79.99%	241.55%
RF	DEF	**0.54**%	9.51%	15.14%
RF	MMS	0.55%	9.54%	**14.97**%
RF	STD	0.55%	**9.50**%	15.23%
RF	OHE	**0.54**%	10.82%	18.32%
SVM	DEF	6.44%	91.46%	249.59%
SVM	MMS	6.07%	91.33%	101.45%
SVM	STD	6.04%	**91.29**%	**100.35**%
SVM	OHE	**6.03**%	91.36%	116.08%
EN	DEF	5.19%	176.36%	246.24%
EN	MMS	5.31%	177.54%	273.19%
EN	STD	5.09%	**169.40**%	267.77%
EN	OHE	**2.54**%	173.59%	**226.60**%

Table 5
Average MAPE value over all tested sample sizes from 50 to 1000 with steps of 50

pattern while being displaced with a relatively constant margin along the y-axis, with STD performing best. The KNN model performed as expected, showing constant improvements with increasing sample sizes for all preprocessing methods. However, it is notable that the best performer on the BerkelyDBC dataset DEF performs the worst now, with the former second-best performing STD taking its place as the best performer. The RF model remains again the best performer with a significant margin. The preprocessing performances are again very similar, but STD performs significantly better for the smallest tested sample size, thus outperforming the others.
Thirdly, we will discuss the largest SPL we investigated, VP9. The results collected for VP9 are shown in Figure 3. Our first observation is that the results for VP9 are closer to the results from BerkeleyDBC. There are again some oscillations in the results of MLP, but they are comparatively minor and show a clear trend to improvement with increasing sample sizes. We see again that the performance of the EN model remains unaffected by increasing sample sizes, except for some minor oscillations. The SVM model shows a similar pattern as it did with the BerkeleyDBC dataset. The DEF preprocessing method performs once more the worst and shows as the only method with no significant improvement with increasing sample size. KNN shows to be once more consistent, showing stable improvement with increasing sample size, STD performing best once more. The RF model performs once more best by a significant margin. The preprocessing methods have little impact on its performance, but some improve the prediction performance earlier, the best performing being MMS.
Finally, we will discuss the results and our observations in general. To this end, we provide the average performances of all models and preprocessing approaches in Table 5. The first observation must be that preprocessing methods have a significant impact on the prediction performances. BerkeleyDBC has an average factor of 2.05 between the best and worst-performing preprocessing methods. In comparison, 7z and VP9 have an average factor of 1.17 and 1.84 respectively. This observation holds for all tested models, even for the best-performer RF. However, RF shows this impact only with the larger SPLs like 7z and VP9. In general, the differences are maximized at low sample sizes and become then smaller with increasing sample sizes. We observe a similar situation with the SVM model, except DEF, which was largely unsuited. DEF was in two out of three tested SPLs performing the worst, showing insignificant improvement with increasing sample sizes. For MLP, KNN, and EN, on the other hand, we can see significant performance differences on every sample size tested, with, in general, more pronounced differences when applying smaller sample sizes. We also observe multiple occasions where misrepresentation of performances could occur when conducting tests with only one preprocessing method. For instance, one can conclude that SVMs outperform MLPs on the BerkeleyDBC dataset for sample sizes smaller or equal to 1000 when conducting tests only with DEF or MMS. However, when testing with STD or OHE, we see that MLP outperforms SVMs on the BerkeleyDBC dataset for sample sizes greater than 650 or 400, respectively. From

this, we conclude that a sound comparison between two or more predictive models should compare their performances when using their best-performing preprocessing methods. Omitting the preprocessing method used may, by extension, lead to poorly reproducible results.

MLP showed to work on average best with OHE. The performance of this model strongly correlated with the sample size, and it usually started with comparatively high MAPE scores that became more competitive with increasing sample sizes. Furthermore, it is prone to oscillation. KNN showed to work on average best with STD. It was one of the most stable and robust models, achieving constant improvement with increasing sample sizes, even in the context of SPLs like 7z that triggered oscillation in most other models. However, its prediction quality places it in the middle field. RF showed to work on average best with MMS. This model outperformed every other model significantly in every aspect we measured. Its worst performance using the smallest tested sample size of 50 outperforms, in all but two cases, the best performances of all other models. This performance is then improved further with increasing sample size. The model usually reaches a plateau relatively early on average at a sample size of 350, after which its improvement slows significantly. SVM showed to work on average best with STD. The model improves like RF on average with a sample size up to 600 steadily, after which the model starts to plateau in its improvement, except for the already mentioned DEF. EN showed to work on average best with OHE. This model showed, on average, comparatively minor improvements with increased sample size.

6. Threats to validity

This paper compared multiple machine learning-based models and explicitly did not perform any parameter tuning for any one of the models. We used, if not stated explicitly differently, always the default parameters defined by the scikit-learn[7] library [14]. Thus, we must acknowledge that fine-tuning the model parameters, especially for the more complex models like MLP, likely will improve the performances of the models employed. However, the measured results are still valid and valuable for comparing the model performances concerning the preprocessing methods and the sizes of the training sets employed.

7. Conclusion

We tested 15 scenarios of machine learning-based performance prediction in the context of SPLs by measuring the performance of five different machine learning models on three SPLs with training sets of increasing sizes. Except for two, all scenarios tested showed, in part, radical changes in prediction quality depending on the preprocessing method used. These changes were most pronounced when we measured the model performances with only a few samples to use as training sets and became less distinctive with training sets of increased size. On average, the disparity between the worst and the best performing preprocessing method were factors of 2.05 (BerkeleyDBC), 1.17 (7z), and 1.84 (VP9). While we identified the on average best performing preprocessing methods for each model we tested, we also see, as visualized in Table 5, that no single method outperforms all others for each dataset, which holds as well if we only focus on a single model. Thus, having shown both the significant impact and the inconsistency in the performance of preprocessing methods, we draw the following conclusions. Results that do not state which, if any, preprocessing method was employed become hard to reproduce. Further, the disregard of preprocessing methods may pose a threat to the validity of results. In summary, preprocessing methods are a high-impact, low-effort, and inconsistent part of the field of SPL performance prediction, and all these properties make them essential to be considered and tested.

References

[1] J. Gong, T. Chen, Deep configuration performance learning: A systematic survey and taxonomy, 2024. arXiv:2403.03322.

[2] P. Clements, L. Northrop, Software product lines, Addison-Wesley Boston, 2002.

[3] E. Engström, P. Runeson, Software product line testing – a systematic mapping study, Information and Software Technology 53 (2011) 2–13. URL: https://www.sciencedirect.com/science/article/pii/S0950584910001709. doi:https://doi.org/10.1016/j.infsof.2010.05.011.

[4] J. Oh, D. Batory, R. Heradio, Finding near-optimal configurations in colossal spaces with statistical guarantees, ACM Trans. Softw. Eng. Methodol. 33 (2023). URL: https://doi.org/10.1145/3611663. doi:10.1145/3611663.

[5] N. Siegmund, M. Rosenmuller, C. Kastner, P. G. Giarrusso, S. Apel, S. S. Kolesnikov, Scalable prediction of non-functional properties in software product lines, in: 2011 15th International Software Product Line Conference, IEEE, 2011, pp. 160–169.

[6] M. F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches,

[7] https://scikit-learn.org/stable/index.html

CoRR abs/1907.06902 (2019). URL: http://arxiv.org/abs/1907.06902. arXiv:1907.06902.

[7] C.-L. Wu, K.-W. Chau, Y.-S. Li, Predicting monthly streamflow using data-driven models coupled with data-preprocessing techniques, Water Resources Research 45 (2009).

[8] J. Rasekhi, M. R. K. Mollaei, M. Bandarabadi, C. A. Teixeira, A. Dourado, Preprocessing effects of 22 linear univariate features on the performance of seizure prediction methods, Journal of Neuroscience Methods 217 (2013) 9–16. URL: https://www.sciencedirect.com/science/article/pii/S0165027013001246. doi:https://doi.org/10.1016/j.jneumeth.2013.03.019.

[9] M. Acher, H. Martin, J. A. Pereira, A. Blouin, J.-M. Jézéquel, D. E. Khelladi, L. Lesoil, O. Barais, Learning very large configuration spaces: What matters for Linux kernel sizes, Ph.D. thesis, Inria Rennes-Bretagne Atlantique, 2019.

[10] J. Guo, D. Yang, N. Siegmund, S. Apel, A. Sarkar, P. Valov, K. Czarnecki, A. Wasowski, H. Yu, Data-efficient performance learning for configurable systems, Empirical Software Engineering 23 (2018) 1826–1867.

[11] H. Martin, M. Acher, J. A. Pereira, L. Lesoil, J.-M. Jézéquel, D. E. Khelladi, Transfer learning across variants and versions: The case of linux kernel size, IEEE Transactions on Software Engineering 48 (2022) 4274–4290. doi:10.1109/TSE.2021.3116768.

[12] J. A. Pereira, M. Acher, H. Martin, J. Jézéquel, G. Botterweck, A. Ventresque, Learning software configuration spaces: A systematic literature review, Journal of Systems and Software 182 (2021) 111044.

[13] J. Alves Pereira, M. Acher, H. Martin, J.-M. Jézéquel, Sampling effect on performance prediction of configurable systems: A case study, in: Proceedings of the ACM/SPEC International Conference on Performance Engineering, ICPE '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 277–288. URL: https://doi.org/10.1145/3358960.3379137. doi:10.1145/3358960.3379137.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[15] M. Acher, H. Martin, L. Lesoil, A. Blouin, J.-M. Jézéquel, D. E. Khelladi, O. Barais, J. A. Pereira, Feature subset selection for learning huge configuration spaces: the case of linux kernel size, in: Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A, SPLC '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 85–96. URL: https://doi.org/10.1145/3546932.3546997. doi:10.1145/3546932.3546997.

[16] L. Bao, X. Liu, F. Wang, B. Fang, Actgan: Automatic configuration tuning for software systems with generative adversarial networks, in: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 465–476. doi:10.1109/ASE.2019.00051.

[17] J. Gong, T. Chen, Predicting software performance with divide-and-learn, in: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 858–870. URL: https://doi.org/10.1145/3611643.3616334. doi:10.1145/3611643.3616334.

[18] S. Fu, S. Gupta, R. Mittal, S. Ratnasamy, On the use of ML for blackbox system performance prediction, in: 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21), USENIX Association, 2021, pp. 763–784. URL: https://www.usenix.org/conference/nsdi21/presentation/fu.

[19] Q. Cao, M.-O. Pun, Y. Chen, Deep learning in network-level performance prediction using cross-layer information, IEEE Transactions on Network Science and Engineering 9 (2022) 2364–2377. doi:10.1109/TNSE.2022.3163274.

[20] J. Cheng, C. Gao, Z. Zheng, Hinnperf: Hierarchical interaction neural network for performance prediction of configurable systems, ACM Trans. Softw. Eng. Methodol. 32 (2023). URL: https://doi.org/10.1145/3528100. doi:10.1145/3528100.

[21] K. Zhu, S. Ying, N. Zhang, D. Zhu, Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network, Journal of Systems and Software 180 (2021) 111026. URL: https://www.sciencedirect.com/science/article/pii/S0164121221001230. doi:https://doi.org/10.1016/j.jss.2021.111026.

[22] D. Nemirovsky, T. Arkose, N. Markovic, M. Nemirovsky, O. Unsal, A. Cristal, A machine learning approach for performance prediction and scheduling on heterogeneous cpus, in: 2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2017, pp. 121–128. doi:10.1109/SBAC-PAD.2017.23.

[23] K.-T. Ding, H.-S. Chen, Y.-L. Pan, H.-H. Chen, Y.-C. Lin, S.-H. Hung, Portable fast platform-aware neural architecture search for edge/mobile computing ai applications, ICSEA 2021 (2021) 108.

[24] Y. Gao, X. Gu, H. Zhang, H. Lin, M. Yang, Runtime performance prediction for deep learning

models with graph neural network, in: 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2023, pp. 368–380. doi:10.1109/ICSE-SEIP58684.2023.00039.

Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems

Lothar Hotz[1,*,†], Christian Bähnisch[1,†], Sebastian Lubos[2], Alexander Felfernig[2], Albert Haag[3] and Johannes Twiefel[4]

[1]*Hamburger Informatik Technologie-Center e.V., Hamburg, Germany*
[2]*Graz University of Technology, Graz, Austria*
[3]*Product Management Haag GmbH, Bad Dürkheim, Germany*
[4]*exXxa GmbH, Hamburg, Germany*

Abstract

Constraint Satisfaction Problems (CSPs) are a core technology that solves many real-world problems, especially for configuration tasks. A key success factor in this context is an efficient knowledge acquisition process where domain experts and knowledge engineers (developers of CSPs) should develop an agreement on the correctness of the expanding knowledge base as soon as possible. In this paper, we show how large language models (LLMs) can be applied to the automated generation of solutions for constraint satisfaction problems thus reducing overheads related to CSP development and maintenance in the future.

Keywords

Constraint Satisfaction Problems, Large Language Models, Knowledge Acquisition, Automated Generation

1. Introduction

Knowledge acquisition for knowledge-based systems, especially constraint-based systems, is a complex task. It includes the formalization of partly tangible knowledge with a knowledge model, such as a configuration, constraint, or feature model [1, 2]. Constrained-based systems, in particular, are often used to implement configuration systems, due to their ability to compute possible values for configurations or even directly configurations themselves [3]. Hence, modeling a configuration problem as a constraint satisfaction problem (CSP) is a typical approach for computing configurations. This type of modeling is similar to developing programs in a high-level programming language but also incorporates logical semantic elements.

Currently in software engineering, the use of large language models (LLMs) is exploited to support programmers in their daily tasks, such as coding, code completion, reviewing code, API programming, generating test cases, documentation, or identifying design patterns as well as learning programming languages or understanding legacy code [4]. This is achieved through general LLMs such as ChatGPT[1] or specific ones like for coding trained LLMs such as CoPilot[2], or software agents like CREWAI[3] and AutoGenStudio[4], or software assisting engineers such as Devine AI[5] or its open source correspondent Devika[6].

An example of developing a small game with a graphical user interface is GPT-Engineer[7], which prompts in natural languages and provides the game logic and the graphical interface through HTML and CSS coding. Especially the combination of an LLM with a compiler or interpreter in a chain leads to a self-evaluation where a code, which is generated code by an LLM, is directly checked through the interpreter, and the resulting error message is taken as the next prompt for the LLM. Through iterating this (e.g., with a supporting tool such as LangCHAIN[8]) a syntactically correct program is created. However, broader tasks, such as reviewing a software architecture or refactoring code distributed over multiple files are hindered by the limited size of a prompt (the context window) to a given number of tokens such as $128,000$ input tokens for GPT-4. In summary, applying LLMs to software development still has to cope with issues such as limited context windows, maintainability

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.
†These authors contributed equally.
lothar.hotz@uni-hamburg.de (L. Hotz); christian.baehnisch@uni-hamburg.de (C. Bähnisch); slubos@ist.tugraz.at (S. Lubos); afelfern@ist.tugraz.at (A. Felfernig); albert@product-management-haag.de (A. Haag); twiefel@exxxa.ai (J. Twiefel)
0000-0001-7370-7726 (L. Hotz)

[1]Chat Generative Pre-Trained Transformer
[2]https://github.com/features/copilot. All URLs accessed in May 2024
[3]https://www.crewai.com/
[4]https://microsoft.github.io/autogen/blog/2023/12/01/AutoGenStudio
[5]https://preview.devin.ai/
[6]https://github.com/stitionai/devika
[7]https://github.com/gpt-engineer-org/gpt-engineer
[8]https://python.langchain.com/docs/get_started/introduction

of AI-created code, and applicability in project planning or software architecture.

Inspired by these approaches, for constraint satisfaction problems the following questions arise:

- Which knowledge acquisition and engineering tasks can be supported by LLMs?
- Can knowledge models be generated by LLMs? How is quality ensured?
- Which alternative implementations of a knowledge model are the most comprehensible or maintainable?

However, a main ingredient of LLMs is of course a prompt that triggers the generation of text, here of knowledge models. Hence, for a knowledge model, the questions are also:

- What type of prompts do we need?
- What has to be presented to the LLM?
- Is natural language text describing the problem enough?
- Should examples of the formal modeling language, which shall be used for formalization, be given?
- Or the complete language specification?

This paper explores the innovative use of LLMs to automatically generate constraint models in constraint programming languages like PyChoco[9]. Constraint programming involves defining problems in terms of variables, related variable domains, and constraints that must be satisfied. Popular problems in this domain include, e.g., the N-queens problem, Magic Square, Map Coloring, and the Traveling Salesperson Problem (TSP). To tackle such problems, we propose a generalized approach (Section 3) where an LLM is leveraged to generate a constraint model based on a provided prompt.

We test our approach based on several constraint satisfaction problem examples (Section 4). In this paper, we mainly concentrate on known examples from the constraint community given by well-known descriptions, not that much on new examples formulated in natural language. Although the task of starting with known problems is untypical for configuration problems, in our view, this step is necessary for exploring the possibilities of LLMs for knowledge modeling in general. This approach of using commonly known constraint and configuration problems builds a baseline for further work.

With our proposed approach, we aim to streamline the process of model creation in constraint programming, making it faster and more accessible by harnessing the capabilities of LLMs. The main idea is to define a general prompt once which describes the problem context and then reuse this general prompt for specific knowledge engineering tasks at hand.

The remainder of this paper is organized as follows. Section 2 provides an overview of the state-of-the-art in LLM-based knowledge modeling. In Section 3, we sketch our proposed LLM-based CSP generation approach. In Section 4, we provide details regarding the LLM prompting approach based on different example CSP tasks. A discussion (Section 5) and a conclusion (Section 6) ends the paper.

2. State of the Art in LLMs and Knowledge Modeling

Many real-world problems can be interpreted as constraint satisfaction problems (CSPs). Knowledge engineers have the task of formalizing domain-specific constraints into corresponding constraint-based representations. This formalization step is often effortful and more assistance and automation are required in the modeling process which can be regarded as a major challenge for constraint-based systems [5]. Due to recent developments in the area of large language models (LLMs), we could envision agent-based approaches that support the (semi-)automated generation of CSPs [6]. In the following, we discuss related work on the application of LLMs to knowledge modeling.

The idea of exploiting synergy potentials of knowledge-based systems and language models for knowledge-base generation has been proposed a.o. by Petroni et al. [7], Ding et al. [8], and Razniewski et al. [9]. Suchanek and Luu [10] motivate the integration of LLMs with data-driven and knowledge-based approaches to exploit the advantages of both worlds, specifically, to let data-driven approaches act as a basis for grounding the LLM output in reality. In the work presented in our paper, a simple form of grounding is the usage of test cases to assure intended CSP semantics. A similar line of research is presented in Nayak and Timmapathini [11] where object and relation identification based on LLMs is investigated.

Ahmed and Choudhury [12] introduce the idea of applying LLMs to the generation of optimization problems. The motivation behind this is to make related problem formulation tasks more accessible to domain experts, i.e., to decrease the need for specialized mathematical knowledge to make problem formulations feasible. Based on a given dataset comprising different example problem definitions (and solutions), LLMs are used to generate optimization problem definitions for new problems defined in a textual fashion. A major difference compared to our work is that model fine-tuning is primarily based on a "human in the loop" approach whereas we focus more on automated fine-tuning giving LLM feedback on the

[9] https://pypi.org/project/pychoco/

syntactical (is the generated CSP correct or what were the compilation issues?) and the semantic level with test cases. A related approach is the application of LLMs to support different types of strategic reasoning tasks, for example, in the context of economic simulations and game theory [13].

A very similar approach provides [14]. This 'Program of Thoughts' named approach also combines an LLM with an external Python interpreter, however, in [14] for computing numerical problems. Furthermore, the verification of the generated Python program is not discussed.

The adaptation of LLMs for logical reasoning tasks [15] can be performed either based on LLM fine-tuning where LLMs are trained (and adapted) for specific problem settings [16] or based on (automated) prompt adaptation [17]. Intending to improve the quality of code generation, Pan et al. [15] follow the idea of using constraint (and SAT) solver execution errors as feedback channels (in the prompting process) to increase the quality of code generation. The feedback approach presented in our paper extends the work of Pan et al. [15] in terms of additionally providing test cases that help to improve the quality of the generated CSP also on the semantic level.

In [18], various prompts are used to create source code with built-in variability. Specific prompts enable the generation of code in different programming languages, allowing for application configuration. However, a human developer would execute these prompts, no iterative process ensures the correctness of the resulting code.

The automated generation of CSPs can also be regarded as a specific type of ontology construction where concept hierarchies are derived based on different LLM prompts representing concept hierarchy-specific queries (see, for example, the work of Funk et al. [19]). An example of such a query is the following: *What are all of the most important subcategories of the category A?*

3. Methodology

Our approach for generating constraint models using LLMs is a structured, iterative process designed to create syntactically and semantically (almost) correct models. This begins by defining a *general LLM prompt* that can encompass the varying specifics of different constraint problems. For instance, the prompt can be tailored to a specific constraint problem but always solicits a PyChoco interpreter that solves the problem.

The process, furthermore, involves using an iterative loop where the LLM generates a constraint model based on the provided prompt. This generated model is then verified through a constraint interpreter or compiler. If the syntax of the model is incorrect, the result of the Python interpreter call is automatically taken as a prompt for a next iteration and the model is adjusted by the LLM and the process repeats. This loop continues until the syntax is correct.

Once a syntactically correct model is produced, the next step is to verify its semantics. This is done by running a series of test cases specific to the problem domain. Successful completion of test cases indicates a correct semantic implementation (at least concerning the test cases).

This approach, thus, executes an iterative prompting of an LLM, where only a human user provides the first prompt, while the results of a Python interpreter are automatically used by the LLM as successive prompts. A complete example of such an iterative dialog including in-between generated answers of the LLM ("Assistant") is presented in the Appendix Section 2.

This methodology can be further detailed through the following steps:

Leveraging Existing Knowledge Problems: To create useful constraint models, we use logical problems that are well understood and documented on the internet. Examples include classic problems such as the N-queens problem, Map Coloring, or the Traveling Salesperson Problem. These problems are selected due to their well-known constraints and solution strategies.

Integration with Constraint Solvers: Once the logical problems are defined, we connect these models to a constraint system, here PyChoco - other suitable solvers could be used. This involves mapping the logical problem's constraints to the syntax and structures of the chosen constraint language and, thus, facilitating automated solving.

Syntax Verification Loop: An iterative process is set up where the LLM generates a constraint model based on the initial prompt. This model is then verified for syntactic correctness using the constraint interpreter or compiler. If errors are identified, the prompt is automatically adjusted by the LLM, by using an error message as a next prompt and the LLM regenerates the model. This loop continues until a syntactically correct model is achieved.

Semantic Verification with Test Cases: After achieving a syntactically correct model, the focus shifts to ensure semantic correctness. This involves introducing test cases specific to the defined problem. The LLM-generated model runs through these test cases to verify that it not only adheres to syntactic rules but also correctly implements the problem's logic. Iteration continues until the model passes all semantic verification tests.

Generalizability: The ultimate goal of our approach is to establish a generalizable method for generating constraint models using LLMs. By systematically using an iterative loop for both syntactic and semantic verification that employs examples and logical problem definitions, we aim to create a robust framework that can be adapted

tool Python_REPL, description: A Python shell. Use this to run and verify your solution to the constraint problem. Input should be a valid Python script using pychoco. The last line should be "verify_solutions(solutions, ...)" in order to verify your solutions
System:
You are an agent designed to solve constraint problems, you write python code to solve and verify constraint problems. You have access to a python REPL, which you can use to execute python code and verify your solution. If you get an error, debug your code and try again. You only use the constraint solver pychoco. You conclude your python code with a variable "solutions" and verify it with the method call "verify_solutions(solutions, <problem_name>", where <problem_name> is given to you as the first word of the instructions. "solutions" should be a list of all solutions to the constraint problem. If verification fails, you correct your solution.
Use the following example as a template for solving constraint problems with pychoco:
User: ints10. find numbers one to 10.
Assistant:

````
```python
from pychoco.model import Model
from test_tool import verify_solutions

model = Model()
x = model.intvar(1, 10, "x")

solver = model.get_solver()
solutions = [s.get_int_val(x) for s in solver.find_all_solutions()]
verify_solutions(solutions, "ints10")
```
````

Figure 1: General prompt including an example for the LLM on how to call the verification method. The first three lines (the tool description) are needed to avoid iterations related to wrongly calling the method `verify_solutions` (see Appendix Section 7 for created errors without those lines).

to various constraint languages and problems.

By following these steps, our approach leverages the capabilities of LLMs to create accurate and efficient constraint models, providing a scalable solution for different problem types.

4. Prompting Constraint Models

We divide prompts into a *general part* which is used with each constraint problem, i.e., the *system prompt* (see Section 4.1), and a *query part* which points to a specific problem (see Section 4.2).

4.1. General Prompts

Prompting Syntactic Valid Models Figure 1 shows the general prompt which we developed for querying for constraint solutions. The prompt provides the following explanations to the LLM:

1. what role it has (“You are an agent…“),
2. the explanation of the REPL[10] as a tool (“You have access…“),
3. a hint of the behavior of the tool (“The python REPL…“),
4. a hint of how to use the tool (“Use this…“),
5. to focus on that tool (“use only“).

Prompting Semantic Verification: To verify, if a semantic correct solution can be computed with the generated constraint model, we include a verification scheme that builds on tests. Those are specific for the constraint problem at hand. However, the general prompt is enhanced with a call to the verification method, which is parameterized with the name of the constraint problem. The verification scheme is given by an example that explains the parameterization, a constraint model, and the call to the verification method (Figure 1).

Prompt Refinement: For more understanding, of why the LLM generates a certain code line in the model, we have enforced the LLM to provide comments above a code line (see Figure 2). This is done by explaining in the prompt what a “Good python code example“[11] is. This is done, by repeating this text at various appropriate

[10]Read-Eval-Print-Loop

[11]The italic style in the prompt is only for the human reader, i.e., only the text is given as prompt.

You are an agent designed to solve constraint problems, you write and execute python code to answer questions. You have access to a python REPL, which you can use to execute python code. The python REPL will keep its state between usage. Use this to gradually approach a final solution. Divide the solution into meaningful parts run them part after part verifying that each part runs correctly. If you get an error, debug your code and try again only the last part that failed. You only use the constraint solver pychoco. Only use the output of your code to answer the question. Write for each code line a comment as needed to justify your reasoning for that code line. I will give you an example of good python code, please, follow the code convention of the ***good python code example***. This ***good python code example***is only an example. This ***good python code example*** has nothing to do with the constraint problem solution. You might know the answer without running any code, but you should still run the code to get the answer. If it does not seem like you can write code to answer the question, just return "I don't know" as the answer. The user may ask questions or give follow up instructions after you presented your solution. You then have to adjust you solution accordingly.
good python code example:

```
"# Import necessary modules
import os

# Define the input and output file paths
input_file_path = \'numbers.txt\'
output_file_path = \'average.txt\'

# Function to read numbers from a file
def read_numbers_from_file(file_path):
    # Initialize an empty list to store the numbers
    numbers = []
    # Open the file in read mode
    with open(file_path, \'r\') as file:
        # Read each line in the file
        for line in file:
            # Strip any leading/trailing whitespace and convert to float
            number = float(line.strip())
            # Append the number to the list
            numbers.append(number)
    # Return the list of numbers
    return numbers
    main()"
```

Figure 2: Refining the prompt to provide comments for the generated model (slightly differently prompted).

positions in the prompt, as well as providing such an example in the prompt.

4.2. Prompting Specific Problems

The query part simply consists of the name of the constraint problem and related specific problem descriptions, see Listing 1[12].

N-Queens Problem

Problem: Positioning n queens with possible movements known from chess on a $n*n$ chessboard[13]. Figure 1 shows the used prompt.

Results: The LLM makes multiple Python REPL calls, one that prints the solution, and one that follows the system prompt on how to verify the solution. This leads to the correct computation of the model including the conversion of the solutions to a list of lists (see Figure 3).

Comments: The solution to the N-queens problem is provided in the PyChoco documentation on the internet, hence, the LLM was trained with it. However, a different solution was found. Also, the 3-queen problem which provides no solution was correctly represented. A further observation was that the result is not deterministic, e.g., it varies arithmetical constraints for diagonals (see Listing 2). Fixing the seed and setting the temperature to zero does not make results deterministic.

The semantic verification test for the N-Queens problem is shown in Listing 3. Through the method

[12]The code in the repository contains further problems such as magic square that lead to similar observations.

[13]See files "chats/queen8_cs_agent_sol2_gpt-4-1106-preview_*" in the repository for results

Listing 1: Queries for specific constraint problems

```
queen3 = "solve the 3-queen problem"
queen8 = "solve the 8-queen problem"
queenn = "solve the n-queen problem, use n=8 as test instance"
---------------------------------------------------------------------------
coloring = """
solve the map coloring problem for four regions, three colors and the given adjacency:
regions = ['A', 'B', 'C', 'D']
adjacency_list = {
     'A': ['B', 'C'],
     'B': ['A', 'C', 'D'],
     'C': ['A', 'B', 'D'],
     'D': ['B', 'C']
}.
the solution should be a list of python dicts where each dict maps regions to color indices
"""
---------------------------------------------------------------------------
tsp = """solve the traveling salesman problem, use the following problem instance:
# Number of cities
C = 4

# Distance matrix
D = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
the solution should be a list of valid solutions, each solution being a list of integers
     representing the cities to be visited.
"""
```

`verify_solutions` called by the prompt, each solution (`queens`) is tested. This test is exhaustive because it analytically tests the queens' position, not leaving out a constraint.

Map Coloring

Problem: Listing 1 describes the well-known Map Coloring as a query.

Results: A first model which contains the constraint `all_different` raised an error[14] but was corrected by using another constraint modeling (i.e., with != instead of `all_different`) through 2 iterations[15]. A further run provides a one-shot success, however not recognized as such, instead a solution was hallucinated for presenting to the user[16]. A further run firstly used the API wrongly, but finds finally a correct solution[17].

Comments: The solution can also be found on the internet, however, this does not prevent from making errors. Finally, also because the iterative approach is used, a solution was found.

Traveling Salesperson Problem

Problem: Listing 1 describes the well-known TSP as a query.

Results: In general, there are several recurring issues when attempting to model constraints for the TSP (see the error logs in Appendix Section 8). One major problem is accurately modeling distance constraints, which often leads to incorrect solutions. Similarly, the Hamiltonian path constraints are difficult to model correctly, compounding the problem. Additionally, there is frequent misuse of the API. These misuses typically involve providing arguments in the wrong order or supplying an incorrect number of arguments. As a result of these issues, the solutions generated are often flawed. With GPT-4, no solutions could be found with the resources at hand, but with GPT-4o, some solutions were correctly identified. However, no implementation was found that computes all the test cases provided for verification. Overall, these challenges underscore the complexity of effectively modeling TSP constraints and highlight areas that require significant improvements to find optimal solutions consistently.

4.3. Implementation

The implementation, which utilizes LangChain and Python, leverages ChatGPT-4 and ChatGPT-4o and is available on a repository at Zenodo[18]. As a constraint

[14]It seems to be an implementation issue, see https://github.com/chocoteam/choco-solver/issues/705
[15]coloring_cs_agent_sol2_gpt-4-1106-preview_ 20240529094847.txt
[16]coloring_cs_agent_sol2_gpt-4-1106-preview_ 20240529095157.txt
[17]coloring_cs_agent_sol2_gpt-4-1106-preview_ 20240529095612.txt

[18]https://doi.org/10.5281/zenodo.12794211

```
Assistant:
There was a mistake in the code. It seems I incorrectly used the
‘solver.find_solution()‘ method, which only finds one solution, and I
didn’t convert the solution to the expected list format before passing
it to the verification function. I need to find all solutions and convert
each solution to a list of integers representing the positions of the
queens.
Let me correct the code.
```

Python_REPL (tool call 1/1):

```
from pychoco.model import Model
from test_tool import verify_solutions
model = Model("8-Queens")
# Create an array of 8 variables representing the positions of queens on a
# chess board
queens = model.intvars(8, 1, 8)
# Add constraints
for i in range(8):
    for j in range(i + 1, 8):
        # Queens cannot be on the same row
        model.arithm(queens[i], "!=" , queens[j]).post()
        # Queens cannot be on the same diagonal
        model.arithm(queens[i], "-", queens[j], "!=", i - j).post()
        model.arithm(queens[i], "-", queens[j], "!=", j - i).post()
# Find all solutions
solver = model.get_solver()
solutions = []
for solution in solver.find_all_solutions():
    solutions.append([solution.get_int_val(queens[i]) for i in range(8)])
verify_solutions(solutions, "queen8")
```

Python_REPL (tool response):
92 of the 92 solutions are correct. Expected 92 solutions

Assistant:
The 8-queens problem has been successfully solved, and all 92 solutions are correct.

tool calls: 6, text messages: 6

Figure 3: Constraint model for 8-queens problem and last output

solver, we used PyChoco as a Python API for the constraint solver Choco[19]. The resources of ChatGPT are limited in respect of context/prompt length. This approach causes the LLM to stop responding when the chat size extends the allowed number of tokens, which easily can occur because the chat includes the entire previous chat history. The resulting chats are included in the paper’s appendix and the repository. The implementation can be used to rerun the experiments, and the data can be utilized for further investigation in prompt engineering.

5. Discussion

Overall, across all problems, after initiating multiple chats, each with several iterations, a model that can compute a single solution could be found (see Appendix sections 3 to 6). In Appendix Section 9, we document the number of iterations per experiment, i.e., calls to the Python interpreter (”Tool Calls”) and answers given by the LLM (”Text Messages”). Nevertheless, for the TSP, we could not find a model that computes all solutions. The iteration was stopped by the LLM, if no changes occurred or if too many tokens were used. So the paper’s result is that in principle constraint models for well-known problems can be computed, however, not in any cases.

An interesting observation is that the generated variable names within the model are always semantically meaningful (e.g., "queens") and not arbitrary. This likely occurs because LLMs statistically favor names that have been previously encountered.

In this paper, we use well-known constraint problems,

[19]https://choco-solver.org/

Listing 2: Non-deterministic results

```
- correct
    model.arithm(queens[i], "!=" , queens[j] + (j - i)).post()
    model.arithm(queens[i], "!=", queens[j] - (j - i)).post()
- correct
    model.arithm(qs[i], "-", qs[j], "!=" , j - i).post()
    model.arithm(qs[j], "-", qs[i], "!=" , j - i).post()
- wrong
    model.abs(queens - queens).ne(i - j).post()  # Diagonal constraint
    model.abs(queens - queens).ne(j - i).post()  # Diagonal constraint
- wrong
    model.arithm(queens[i], "-", queens[j], "!=", i - j).post()
    model.arithm(queens[i], "+", queens[j], "!=", i + j + 2).post()
- does not find all solutions
    diag1 = [model.int_offset_view(queens[i], i) for i in range(n)]
    diag2 = [model.int_offset_view(queens[i], -i) for i in range(n)]
    model.all_different(diag1).post()
    model.all_different(diag2).post()
- wrong
    model.arithm(queens[i], "!=" , queens[j]).post()  # Different columns
    model.arithm(queens[i], "-", i, "!=" , queens[j], "-", j).post()  # Different diagonals (left-
         top to right-bottom)
    model.arithm(queens[i], "+", i, "!=" , queens[j], "+", j).post()  # Different diagonals (left-
         bottom to right-top)
- wrong
    model.all_different([queens[i], queens[j]]).post()
    model.arithm(queens[i], "-", queens[j], "!=" , j - i).post()
    model.arithm(queens[i], "+", queens[j], "!=" , j - i).post()
```

Listing 3: Verifying a solutions for N-queens through testing the queens' positions.

```python
def is_valid_solution(queens):
    n = len(queens)
    assert type(queens) == list, "expected list of ints"
    for i in range(n):
        for j in range(i + 1, n):
            # Check if queens are in the same row
            if queens[i] == queens[j]:
                return False
            # Check diagonals: if the difference between the column indices equals
            # the difference between the row indices, they're in the same diagonal
            if abs(queens[i] - queens[j]) == abs(i - j):
                return False
    return True
```

which, of course, can only be a first step for leveraging LLMs in knowledge acquisition. The next steps would be to supply new, unknown problems, also industrial ones, e.g., by giving tables of correct variable combinations (configurations) in table constraints and generating abstracted constraint models. However, Listing 1 shows a potential way of representing formally a problem as a query which probably can be used for other tasks.

There are several general problems identified: the program persistently uses the PyChoco API incorrectly and often hallucinates PyChoco functions. This could not be prevented by providing the API definition in the prompt, because the complete definition extends the number of possible tokens we could send to the LLM.

Additionally, the chats are not stable and can vary significantly even with the same prompt. This instability persists even with temperature set to 0 and a fixed seed, making chats non-deterministic.

The used prompts should be further developed to achieve the intended general generation of knowledge models; in other words, prompt engineering for knowledge modeling needs to be elaborated. Few-Shot Prompting [20] could be used to encourage the model to produce more consistent outputs and adhere more closely to given instructions.

The approach of first creating syntactically correct models and then verifying them to ensure semantic accuracy could be successfully demonstrated (see Listing 3). Because we use a test-driven semantic verification this verification depends on the quality of the used test cases. For example, the N-queen problem was exhaustive because of an analytical-based verification and not an enumeration of test cases. Furthermore, writing tests for knowledge-based tasks is, in our experience, typically easier than writing knowledge models, hence, the experts' tasks of creating formal constraints and also configuration models will, in our opinion, shift from writing models to writing tests - and let us write the models through an LLM. Thus, the knowledge model developing process probably, with the presented approach, can be executed by domain experts not only by knowledge engineers.

The paper offers, furthermore, an easy way to begin with iterative LLM prompting. Moreover, it is suggested to use the LLM for creating a new configuration language by exploiting the hallucination capabilities.

6. Conclusions

The paper presents the first steps in supporting the formalization task in constraint modeling with LLMs, i.e., the mapping of a well-known problem into a constraint representation of a certain constraint language, here PyChoco, with large language models. By taking various constraint problems, formulating generally applicable prompts, and doing so in an iterative manner which includes syntactic and semantic verification processes, the LLM could generate appropriate and correct constraint models. Through the iteration, in-between errors related to syntax and semantics were automatically corrected. Further work will include a quantitative evaluation of the method, a comparison of different LLMs, as well as the generation of constraint models for unknown problems. In total, this paper provides a first step into a new kind of constraint and configuration modeling approach with LLMs.

Acknowledgments

This project was funded through the Federal Institute for Research on Building, Urban Affairs and Spatial Development (Bundesinstitut für Bau-, Stadt- und Raumforschung), Germany, within the Zukunft Bau Forschungsförderung Maßnahme (Future of construction Research funding Measure): "AI-supported modernization of rental housing stock" ("KI-gestützte Modernisierung an Mietwohnungsbaubeständen (KIMM)") as well as through the Federal Ministry for Economic Affairs and Climate Action (Bundesministerium für Wirtschaft und Klimaschutz), Germany, within the "Joint project: EnOB: IntelMOD - Intelligent modernization platform based on functional cost splitting; sub-project: Infrastructure of the modernization platform." ("Verbundvorhaben: EnOB: IntelMOD - Intelligente Modernisierungsplattform auf Basis des Funktionalen Kostensplittings; Teilprojekt: Infrastruktur der Modernisierungsplattform.")

References

[1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann, 2014.

[2] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-driven Design, Analysis, and Applications, Springer, 2024.

[3] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, Configuration Knowledge Representation & Reasoning, in: A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann Publishers, 2014, pp. 59–96.

[4] I. Ozkaya, A. Carleton, J. Robert, D. Schmidt, Application of large language models (llms) in software engineering: Overblown hype or disruptive change?, Carnegie Mellon University, Software Engineering Institute's Insights (blog), 2023. URL: https://doi.org/10.58012/6n1p-pw64, accessed: 2024-May-16.

[5] B. O'Sullivan, Automated modelling and solving in constraint programming, in: 24th AAAI Conference on Artificial Intelligence, AAAI'10, AAAI Press, 2010, pp. 1493–1497.

[6] E. C. Freuder, Conversational modeling for constraint satisfaction, AAAI Conference on Artificial Intelligence 38 (2024) 22592–22597. URL: https://ojs.aaai.org/index.php/AAAI/article/view/30268. doi:10.1609/aaai.v38i20.30268.

[7] F. Petroni, T. Rocktäschel, P. P. Lewis, A. Bakhtin, Y. Wu, A. Miller, S. Riedel, Language models as knowledge bases?, 2019. URL: https://arxiv.org/pdf/1909.01066.

[8] Y. Ding, A. Poudel, Q. Zeng, T. Weninger, B. Veeramani, S. Bhattacharya, Entgpt: Linking generative large language models with knowledge bases, 2024. URL: https://arxiv.org/abs/2402.06738.

[9] S. Razniewski, A. Yates, N. Kassner, G. Weikum, Language models as or for knowledge bases, 2021. URL: https://arxiv.org/abs/2110.04888.

[10] F. Suchanek, A. T. Luu, Knowledge bases and language models: Complementing forces, in: 7th Inter-

national Joint Conference on Rules and Reasoning, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 3–15. URL: https://doi.org/10.1007/978-3-031-45072-3_1.
[11] A. Nayak, H. P. Timmapathini, Llm2kb: Constructing knowledge bases using instruction tuned context aware large language models, 2023. URL: https://arxiv.org/abs/2308.13207.
[12] T. Ahmed, S. Choudhury, Lm4opt: Unveiling the potential of large language models in formulating mathematical optimization problems, 2024. URL: https://arxiv.org/abs/2403.01342.
[13] Y. Zhang, S. Mao, T. Ge, X. Wang, A. de Wynter, LLM as a mastermind: A survey of strategic reasoning with large language models, 2023. URL: https://arxiv.org/abs/2404.01230v1. arXiv:2404.01230.
[14] W. Chen, X. Ma, X. Wang, W. W. Cohen, Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, Transactions on Machine Learning Research (2023). URL: https://openreview.net/forum?id=YfZ4ZPt8zd.
[15] L. Pan, A. Albalak, X. Wang, W. Y. Wang, Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning, 2023. arXiv:2305.12295.
[16] P. Clark, O. Tafjord, K. Richardson, Transformers as soft reasoners over language, in: 29th International Joint Conference on Artificial Intelligence, IJCAI'20, 2021, pp. 3882–3890.
[17] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. hsin Chi, D. Zhou, Self-consistency improves chain of thought reasoning in language models, ArXiv (2022). URL: https://arxiv.org/abs/2203.11171.
[18] M. Acher, J. G. Duarte, J.-M. Jézéquel, On programming variability with large language model-based assistant, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 8–14. URL: https://doi.org/10.1145/3579027.3608972.
[19] M. Funk, S. Hosemann, J. C. Jung, C. Lutz, Towards ontology construction with language models, 2023. URL: https://arxiv.org/abs/2309.09898.
[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

Configuration Copilot: Towards Integrating Large Language Models and Constraints

Philipp Kogler[1,*], Wei Chen[1], Andreas Falkner[1], Alois Haselböck[1] and Stefan Wallner[1]

[1]Siemens AG Österreich, Siemensstraße 90, 1210 Wien, Austria

Abstract

A product configurator enables the configuration of a customizable product while constraining possible variations. Users typically interact with a product configurator via a graphical user interface. A complex product can be composed of components and parameters that are not easily understandable for non-experts which can prevent them from effectively configuring the product. In this paper, we propose a configuration copilot, an interactive chat-based interface that allows users to iteratively configure a product by describing their requirements in natural language. Our framework leverages the Natural Language Processing (NLP) capabilities of advanced pre-trained Large Language Models (LLMs) alongside the robustness of constraint-based product configurators. We introduce a technical architecture that accurately formalizes constraints from natural language inputs, identifies valid product configurations based on a defined product line and specified constraints using a constraint solver, and communicates the resulting product configurations back to the end user in natural language. We demonstrate and evaluate the configuration copilot on two use-cases: The configuration of the GoPhone feature model (Boolean feature assignments), and the configuration of a metro wagon (more general configuration parameters).

Keywords

Product Configuration, Constraints, Feature Models, Large Language Models, Copilot

1. Introduction

Product configuration involves creating customized products from predefined components while satisfying constraints that limit configurable parameters and possible combinations [1]. A product configurator is a software tool that allows users to configure a product, commonly through a graphical user interface and often in a web-based context. Therefore, interface and interaction design plays a major role in the development of a product configurator but is often overlooked [2]. This observation is especially relevant when complex products are configured by non-expert users. The meaning of configurable components and parameters may not be obvious which prompts a need for explanation and introduces a learning curve.

As an alternative to GUI-based interactions with product configurators, we propose a configuration copilot that offers a text-based chat interface. Uninformed users shall be able to describe their requirements in natural language without knowledge of the concrete parameters to set and components to select. The copilot shall then configure the product and respond with a valid configuration complying with the initial requirements. The user shall be able to interactively refine the product configuration.

We utilize a pre-trained Large Language Model (LLM) for the processing of natural language. Recent advances in this field have enabled use cases that require the understanding and generation of not only natural language but also code. Well-known limitations include a lack of reliability, guaranteed correctness, domain-specific knowledge in general-purpose LLMs, and limited reasoning abilities [3]. In our configuration copilot, we address these shortcomings by combining a LLM with a constraint solver. While the strengths of the LLM are utilized in the processing of the natural-language requirement descriptions, the reasoning to find valid configurations is done by the constraint solver.

In this paper, we first describe LLMs and constraint-based product configuration in Section 2 and related work in Section 3. We detail the technical architecture of the configuration copilot in Section 4, and present an evaluation based on the two use-cases of configuring the GoPhone feature model and a metro wagon in Section 5. We conclude the paper with a summary, a limitation statement, and future work in Section 6.

2. Background

2.1. Large Language Models

Pre-training task-agnostic aspects of natural language processing (NLP) tasks is a central concept of LLMs. The *Transformer* architecture enables this approach on a large scale through parallelization. Transformer

ConfWS'24: 26th International Workshop on Configuration, Sep 2–3, 2024, Girona, Spain

*Corresponding author.
philipp.kogler@siemens.com (P. Kogler); chen.wei@siemens.com (W. Chen); andreas.a.falkner@siemens.com (A. Falkner); alois.haselboeck@siemens.com (A. Haselböck); stefan.wallner@siemens.com (S. Wallner)
0009-0009-5598-1225 (P. Kogler); 0009-0008-0486-9068 (W. Chen); 0000-0002-2894-3284 (A. Falkner); 0000-0003-2599-3902 (A. Haselböck); 0000-0002-9755-6632 (S. Wallner)

models are able to capture complex patterns and long-range-dependencies in texts through the multi-head self-attention mechanism. Compared to previous state-of-the-art models such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) a performance improvement in various NLP tasks is observed [4, 5].

Decoder-only models are a subclass of Transformer-based architectures and are primarily used for sequence-to-sequence tasks such as translation. Auto-regressive models predict the next single token (sub-word) by maximizing the log-likelihood given all previous words and the model parameters [4].

The size and quality of the pre-training corpus have a strong impact on performance [5]. LLMs are trained on publicly available data and excel in general language tasks. Highly specialized tasks require expert knowledge that is often not included in the training data, and therefore LLMs may not be able to generate accurate output. Task-specific knowledge can be introduced to a general-purpose LLM through domain customization by employing techniques like *prompting* and *fine-tuning* [6].

2.2. Constraint-based Product Configuration

Product configuration involves selecting and assembling various components and options to meet customer requirements and constraints. Its complexity arises from the vast number of possible combinations and the need to satisfy all technical restrictions and customer preferences. To handle this complexity, powerful technologies have been developed and established in the last decades. Constraint-based systems shall be highlighted here, which allow to represent the product line and its technical restrictions and requirements in a clean, logical way, thereby ensuring that only valid configurations are generated. The core of such systems lies in the ability to handle complex and combinatorial search spaces efficiently through the use of advanced solving algorithms, such as backtracking, forward checking, and constraint propagation. This facilitates the efficient generation of feasible solutions while pruning invalid combinations.

An important subdomain of configuration problems are feature models for the representation of product lines [7]. Constraint-based techniques are especially well-suited for such feature models, because of the simple language and the mainly Boolean type of the variables.

MiniZinc is a constraint language that can be used to represent configuration problems [8]. Several efficient solvers can process this language and can therefore be used as the backend of a configurator.

A product configurator is almost always an interactive system [9]. A graphical user-interface (GUI) allows to enter the user requirements, which are passed on as input to the constraint solver. The results of the solver are presented on the GUI, and the user can vary or refine her/his input specification and the solver is called again.

To design and implement a configurator GUI can be a challenging task, because the possible interactions are diverse, like collecting the requirements, reporting invalid constellations, representing a solution, showing a performance value of a solution, etc. In addition, every modification of the product (line) requires a review and possibly and adjustment of the GUI.

In the following sections, we demonstrate how to eliminate the need for a product-specific GUI by utilizing an LLM to engage in dialogue with the user.

3. Related Work

Various approaches to improve the reliability, the performance in domain-specific tasks, and the reasoning abilities of LLMs are described in literature.

Few-shot prompting effectively introduces domain-specific knowledge and improves the task-specific performance of LLMs by adding a small set of example interactions (input and expected output) to the prompt [10]. *Chain-of-thought prompting* was shown to improve the reasoning abilities of LLMs especially in more complex tasks by providing exemplary intermediate reasoning steps [11].

Grammar prompting is used when a specific output format is expected. Wang et al. describe how a minimal specialized grammar is obtained in a grammar specialization process by selecting a specialized grammar as a subset of the full grammar using an LLM and minimizing it by parsing the output and forming the union of used rules. In their approach, constrained decoding then validates the output syntax [12]. Similarly, Poesia et al. presented the *Synchromesh* framework: Using a few-shot prompting technique, semantically similar examples are selected from a larger pool for a given natural language prompt via a similarity metric named *Target Similarity Tuning*. Constraints are enforced through *Constrained Semantic Decoding* to verify syntax validity, scoping, or type checks. During the token-by-token construction of the LLM output, a *Completion Engine* provides all valid tokens that can further extend a partial program towards a full correct program [13].

Neuro-symbolic approaches focus on combining the strengths of neural networks and symbolic reasoners. Pan et al. introduced the *Logic-LM* framework which achieves a performance improvement of 18% on logical reasoning datasets over chain-of-though prompting. The framework translates the natural-language input into symbolic formulations and utilizes a symbolic reasoner to obtain the answer [14].

This paper builds upon our previous work [15] that studied the reliable generation of formal specifications with LLMs using algorithmic post-processing. We extend the approach towards product configuration by applying post-processing to reliably integrate a constraint solver. In addition to previously described guaranteed syntactically valid output, this extension enables arbitrary semantic constraints.

4. Configuration Copilot

This section presents the technical details of the configuration copilot that combines LLMs with constraint-based configuration.

4.1. Architecture

Figure 1 shows an overview of the architecture. A user configures a product by providing a natural-language description of their requirements. The Formalizer (see Section 4.2) is a specialized LLM-based component that translates the requirements to constraints. The Configuration Engine is a constraint solver that attempts to find a configuration that satisfies the general constraints of the product line combined with the user constraints provided by the Formalizer. An Interpreter (see Section 4.4) translates the configuration back to natural language. The configuration copilot then responds with a natural-language description of the configured product accompanied by the full technical specification (product configuration as determined by the Configuration Engine). The user can then further refine the product configuration interactively.

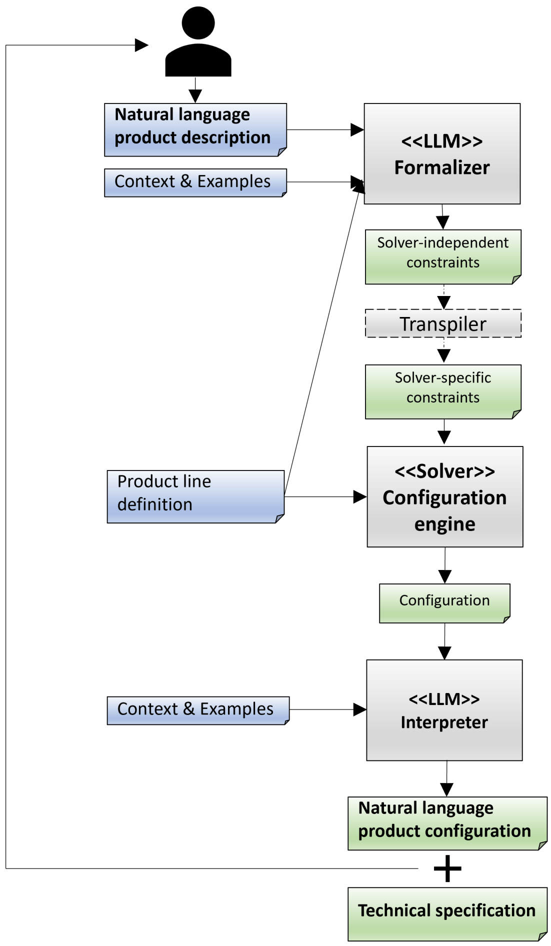


Figure 1: Architecture of the Configuration Copilot

4.2. Formalizer

The input to the Formalizer is a natural-language description of arbitrary product requirements provided by a non-expert. Utilizing the NLP capabilities of LLMs, the formalization can be viewed as a sequence-to-sequence translation task from natural language to a formal specification. The LLM is tasked with natural language understanding and the identification of corresponding parameters or components of the product (line), but is specifically not tasked with reasoning (e.g., constraint satisfaction). While pre-trained LLMs achieve a strong performance on general tasks, they do not have knowledge of the specific product (line) to configure as corresponding data is not included in their training corpus [5, 6]. Additionally, the probabilistic nature of the token-by-token output construction of LLMs does not provide any guarantees in the correct generation of valid constraints [4].

Our framework for reliable code generation addresses domain-customization and reliable output generation through few-shot prompting and algorithmic post-processing [15].

Few-shot prompting has been shown to effectively extend the capabilities of LLMs with domain knowledge while requiring significantly less training data than fine-tuning [10]. Knowledge of the product line is incorporated through a system prompt describing the product line with its parameters and components. A small set of examples is appended as pairs of natural-language inputs and expected outputs to provide the LLM with more context and guide it towards the expected behavior.

Rather than generating output directly in a specific constraint language, an intermediary JSON-based language is used, which can then be easily transpiled. The transpiler parses the JSON constraint representation and maps its elements to corresponding constructs of the specific constraint language following predefined rules. As JSON is widely used, pre-trained LLMs have more often encountered JSON than less common constraint languages. Therefore, the generation of an intermediary JSON output is closer to the LLMs capabilities. Additionally, an intermediary language gives more control over the expected output as available language constructs can be constrained and tailored to the specific task. It also decouples the Formalizer from the Configuration Engine by enabling interchangeability of the concrete constraint language. To generate a valid JSON for the Formalizer, several state-of-the-art LLMs are evaluated and benchmarked. Specialized code LLMs that are pre-trained on the translation of natural language to code in a variety of programming languages are believed to be more suitable for the generation of structured JSON output. In our evaluation in Section 5, we selected four open-access LLMs: Two code LLMs (CodeLLama [16] and Codestral [17]), and two general-purpose instruction-tuned LLMs (Meta Llama 3 [18] and Mistral [19]).

Algorithmic post-processing guarantees the correct generation of the JSON-based intermediary language and is depicted in Figure 2. As the auto-regressive Transformer model generates its output step-by-step as tokens, the post-processor engages into every generation step: For each step, the LLM generates a list of candidates for the next token based on the prompt and the generated output so far. Sorted by priority as evaluated by the LLM, the post-processor determines whether the token candidate represents a valid continuation of the partial output sequence (partial intermediary JSON). The valid token candidate with the highest priority is then selected, handed back to the LLM, and added to the partial JSON, extending it one step further. A completeness checker evaluates after every step to determine, if the JSON is complete [15].

The JSON-based intermediary language is formally defined by a JSON schema specification and the post-processor is therefore a specialized JSON validator that can strictly validate any partial JSON against the schema. This implementation is based on deterministic finite automata (DFA). Each generic JSON language element (object, list, string, number, etc.) is represented by a DFA, keeping track of the current state. The token generated by the LLM is broken down to single-character inputs for the JSON validator. Depending on the schema and the current state, only a set of characters is accepted. If a character is rejected, the current token is considered invalid, and the validator state is rolled back to the last valid token. State changes are triggered by characters until the final state is reached. When the DFA reaches its final state, the generated valid JSON is complete [15].

4.3. Configuration Engine

Given the user constraints combined with the complete product line definition, the Configuration Engine evaluates whether the constraints are satisfiable and returns a configuration. The product line as well as the user constraints are modelled in the MiniZinc constraint language [8]. The solver returns the full product configuration as a list of variable assignments which serves as an input to the Interpreter. In this context, we consider the constraint solver a given technology that will neither be further described nor evaluated.

4.4. Interpreter

The Interpreter is an LLM module that explains the product configuration found by the Configuration Engine. The goal is to provide the user with a less technical summary that is understandable for non-experts.

Structured few-shot prompting [10] is sufficient for this use-case as LLMs generally perform well in the translation from a formal specification to a natural-language summary as all facts are directly present in the prompt. The context given to the LLM consists of three aspects: The product line definition, instructions, and examples. The LLM is prompted to evaluate which properties and components are most important to be included in the summary. This is achieved by adding importance hints to the product line definition, and by appending the original user input. Properties and components mentioned directly in the user input are given more importance and are more likely to be included in the summary. The result is a more natural context-aware explanation of the most relevant aspects in the product configuration.

5. Evaluation

The presented configuration copilot is evaluated on two use-cases: The conceptually simpler task of configuring a feature model, and the configuration of a metro Wagon.

5.1. Feature Model (GoPhone)

The first use-case for the evaluation of the presented copilot is the configuration of a feature model. An uninformed user shall be supported in the configuration of the GoPhone from the SPLOT project [20].

The GoPhone is a feature model comprised of 77 features with some being mandatory, optional, dependent on other features, or mutually exclusive. For example, the feature `call` is mandatory for the GoPhone, the feature `accept_incoming_call` is mandatory for `call`, but `show_missed_calls` and `show_received_calls` are optional.

Feature assignments are Boolean, either the feature is included in the product configuration (`true`) or the feature is not included (`false`). The product line definition is a MiniZinc program that was directly derived from the

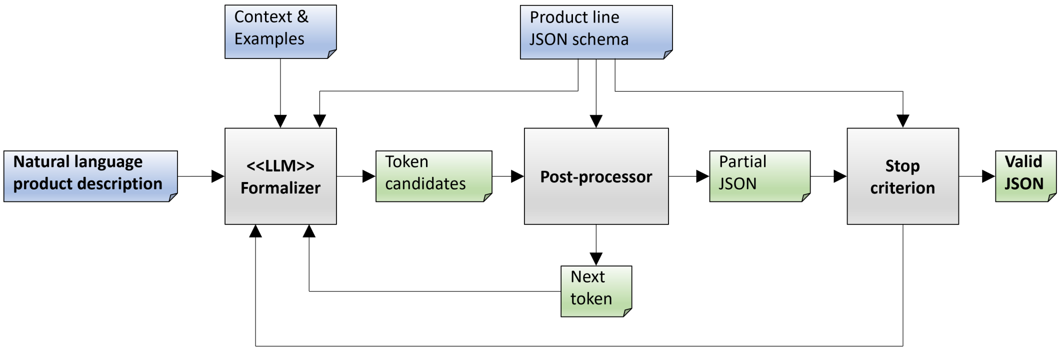


Figure 2: Detail view of the Formalizer with post-processing

feature model. Each feature is a Boolean variable. Constraints limit the combination of features and therefore limit possible product configurations.

A non-expert user starts by describing their requirements for the phone in natural language:

```
I need a basic phone to call people and browse
the web but I don't play games. I also want to
keep track of my appointments.
```

This natural-language description is then formalized to the intermediary JSON language:

```
{
    "features": [
        {
            "name": "make_call",
            "value": true
        },
        {
            "name": "browsing",
            "value": true
        },
        {
            "name": "game",
            "value": false
        },
        {
            "name": "calendar_entry",
            "value": true
        }
    ]
}
```

This list of solver-independent constraints is then transpiled to MiniZinc constraints:

```
constraint make_call = true;
constraint browsing = true;
constraint game = false;
constraint calendar_entry = true;
```

Together with the MiniZinc program (product line definition), the Configuration Engine evaluates the constraints and returns a full product configuration of the GoPhone for the specific user requirements as a list of Boolean feature assignments. In this work, the Gecode [21] solver was used without further configuration or optimization. The Interpreter converts this configuration back to natural language and returns it to the user. An example for such an output is (the technical specification is shortened for brevity):

```
Your GoPhone can manage ringing tones, messages,
and browse the web. It can also manage calls,
read multimedia, and display photos. It has a
calendar entry feature and an address book
processing system. However, it does not play
games, organize tasks, or have currency
conversion features.

Here is the full technical configuration:

GoPhone = true;
manage_ringing_tones = true;
[...]
browse = true;
[...]
game = false;
play_games = false;
install_games = false;
[...]
```

The crucial and potentially failing component of the presented architecture is the Formalizer: the probabilistic nature of the underlying LLMs does not provide strict guarantees. Especially the translation of the user's requirements to the feature assignments is subject to uncertainty. A formal evaluation of the Interpreter is not done because the correctness requirements for the configuration summary are less strong and LLMs are generally known to perform well on simple summarization tasks when the facts are directly provided. It is also unsuitable to define a single reference solution as a large variety of summaries (with various feature assignments being explained or not explained) could be considered correct. Ultimately, users needs to decide whether the summary was helpful or not.

The Formalizer was evaluated on a custom dataset of 30 test cases. 15 test cases create a new configuration from scratch, and 15 test cases evaluate a re-configuration where a given configuration is modified. Each test case consists of natural-language input mentioning between two and six feature requirements in the text (and up to 30 given feature assignments for modification test cases), and the expected feature assignments in JSON. Using the natural-language input, the Formalizer generates feature assignments in JSON. This output is compared to the expected output. The comparison is conceptually challenging due to the intrinsic ambiguity of natural language. In many cases, one could argue for multiple options of feature assignments to be considered a correct translation. In this evaluation, we hand-crafted the dataset to be less ambiguous. However, the features of the GoPhone are in themselves sometimes not obviously distinguishable, and multiple features may be equally suitable. For example, the feature `browsing` is an optional sub-feature of the more general parent feature `browse`. This ambiguity was addressed by encoding very similar features to the same representation. Therefore, all defined synonymous features are considered a correct feature assignment for a requirement. However, the feature assignment was not limited to leaf features because doing so would add reasoning requirements to the Formalizer. Consider the leaf features `play_games` and `install_games`, and the parent feature `game`. If a user only mentions games in their descriptions, the more abstract feature `game` shall be assigned. Otherwise, the LLM would have to reason about a proper assignment of leaf features, deviating from the most direct translation from natural language to a feature assignment. The reasoning regarding further (sub-)feature assignments shall be done by the Configuration Engine.

A similarity metric based on the Jaccard distance between sets [22] was used to compare each pair of expected and actual output: Let T and F be the sets of feature names in the expected output where the feature value is *True* and *False*, respectively. Similarly, let $\hat{T}$ and $\hat{F}$ be the sets of feature names in the actual output where the feature value is *True* and *False*, respectively. The Jaccard similarities are:

For the *True* sets:

$$S_T = \frac{|T \cap \hat{T}|}{|T \cup \hat{T}|}$$

For the *False* sets:

$$S_F = \frac{|F \cap \hat{F}|}{|F \cup \hat{F}|}$$

Table 1
Evaluation Results for the GoPhone Formalization
S = Similarity score
F1 = F1 score

Model [Size/Quantization]	S	F1
CodeLlama 34B/Q4 [Link]	0.65	0.74
Codestral 22B/Q4 [Link]	**0.79**	**0.86**
Meta Llama 3 8B/Q8 [Link]	0.46	0.58
Mistral 7B/Q8 [Link]	0.69	0.79

The overall similarity between the expected and actual output S as the weighted average of S_T and S_F is:

$$S = \frac{S_T \cdot |T \cup \hat{T}| + S_F \cdot |F \cup \hat{F}|}{|T \cup \hat{T}| + |F \cup \hat{F}|}$$

The result is a number between 0 and 1, with 0 indicating no similarity, and 1 indicating a perfect match. In this metric, the identification of features in the natural language as well as the Boolean assignment are considered.

Similarly, the precision P, recall R and F1 score $F1$ were calculated:

$$P = \frac{|T \cap \hat{T}| + |F \cap \hat{F}|}{|\hat{T}| + |\hat{F}|}$$

$$R = \frac{|T \cap \hat{T}| + |F \cap \hat{F}|}{|T| + |F|}$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

We selected four open-access LLMs from HuggingFace to be evaluated in the context of the configuration copilot, two code models, and two general-purpose models. Table 1 summarizes the evaluation results for the GoPhone use-case per LLM. Codestral 22B/Q4, a state-of-the-art code model, performed best. However, Mistral 7B/Q8 outperformed the larger code model CodeLLama 34B/Q4 against our expectations. This shows that the performance of LLMs is use-case specific and must be evaluated. We found that the performance degrades as instances become more complex. Remedies for this observation are the use of larger models, tuning the technical approach, or future improvements of LLMs themselves. Considering the remaining ambiguity of natural language, the results indicate reasonable performance in this use-case as the majority of feature requirements was formalized correctly.

5.2. Metro Wagon

The second use-case for the evaluation is a metro Wagon configuration problem (see [23]) that uses not only Boolean but also numeric variables and arrays, where a configurable product has components that can occur

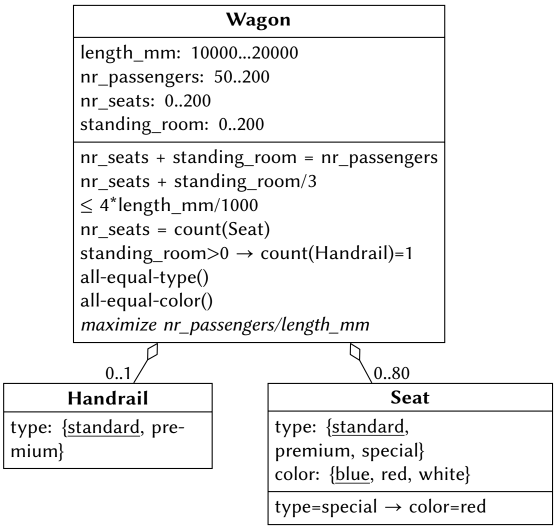


Figure 3: Class diagram of the Wagon example. *Default values are* <u>*underlined*</u>*. Wagon.all-equal-type() stands for a constraint that all sub-parts must have the same type except for special. Wagon.all-equal-color() stands for a constraint that all associated seats (except if type=special) must have the same color.*

multiple times (similar to generative constraint satisfaction [24] or cardinality-based feature modelling [25]).

A metro train wagon has as configurable attributes the size (length in millimetres: 10000..20000) and the expected load (number of passengers: 50..200) which can be realized as seats or standing room. As components we consider only seats (max. 4 per meter of length) and handrails, and their number is configurable.

There is at most one handrail in a wagon (mandatory if there is standing room) and it has a configurable type: "standard" or "premium".

A single seat consumes standing room for 3 persons and has as configurable attributes the type ("standard", "premium", "special") and the color ("blue", "red", "white"). The type is constrained such that standard is not allowed to be mixed with premium (for seats and handrails). The color of all seats must be the same, except for special seats which have to be "red".

Figure 3 shows a UML class diagram for this sample specification, including pseudo code for all constraints.

A non-expert user starts by describing their requirements for the metro Wagon in natural language:

```
The wagon should accommodate more than 120 people
 with room for 40 to sit. Seats should be red.
```

This natural-language description is then formalized to the intermediary JSON language:


```
{
   "nr_passengers": {
       "type": "greaterThan",
       "value": 120
   },
   "nr_seats": {
       "type": "equals",
       "value": 40
   },
   "seat_color": [
       "red",
       "red",
       "red",
       ...
   ]
}
```


This list of solver-independent constraints is then transpiled to MiniZinc constraints:

```
constraint nr_passengers > 120;
constraint nr_seats = 40;
constraint forall (i in 1..nr_seats)
   (seat_color[i] = red);
```

Together with the MiniZinc program (product line definition), the Configuration Engine evaluates the constraints and returns a full product configuration of the metro Wagon for the specific user requirements as a list of value assignments to the configurable parameters. The Interpreter converts this configuration back to natural language and returns it to the user:

```
Your metro Wagon is 20 meters long, has space for
 160 passengers with 40 red standard seats and a
standard handrail. There is also standing room
for an additional 120 people.

Here is the full technical configuration:

length_mm = 20000;
nr_passengers = 160;
nr_seats = 40;
standing_room = 120;
nr_handrails = 1;
handrail_type = standard;
seat_color = [red, red, red, ...];
seat_type = [standard, standard, standard, ...];
```

The Formalizer for the metro Wagon use-case was evaluated, like the GoPhone Formalizer, on a diverse set of 30 test cases (pairs of input and expected output) with 15 creating a new configuration and 15 modifying a given configuration (re-configuration). To evaluate the similarity in this use-case, the previously described similarity metric based on the Jaccard distance between sets for Boolean feature assignments was extended to the more general use-case. This extension is necessary to enable the evaluation of the value assignment for the

Table 2
Evaluation Results for the Metro Wagon Formalization
S = Similarity score

Model [Size/Quantization]	S
CodeLlama 34B/Q4 [Link]	0.77
Codestral 22B/Q4 [Link]	**0.78**
Meta Llama 3 8B/Q8 [Link]	0.68
Mistral 7B/Q8 [Link]	0.72

extended variable types (i.e., strings, numbers, arrays). While the Jaccard distance remained the basis for the similarity metric, a type-specific value metric was applied to each configuration parameter that is present in both, the expected and the actual output. In addition to the parameters being present, the total similarity is adjusted according to the value similarities as well. The type-specific metric considers:

- for numeric values: operator ('=', '>', '<', etc.) and value distance relative to the parameter-specific domain (value range)
- for array values: length and positional item equality
- for string-enumerated values: exact value match

Let C be the set of configuration parameter names in the expected output and let $\hat{C}$ be the set of configuration parameter names in the actual output. The Jaccard similarity S_J is:

$$S_J = \frac{|C \cap \hat{C}|}{|C \cup \hat{C}|}$$

Let c be a matching parameter that is in both, the expected and the actual output, and let $S_v(c)$ be the type-specific value similarity (between 0 and 1) of c between the expected and the actual output. The value-adjusted Jaccard similarity S is then:

$$S = \frac{\sum_{c \in C \cap \hat{C}} S_v(c)}{|C \cup \hat{C}|}$$

The evaluation of the F1 score is omitted because it does not provide any additional value, as it appears to correlate strongly with the already rather strict similarity score S. Table 2 summarizes the evaluation results for the metro Wagon use-case per LLM. Codestral 22B/Q4 performed best again with a similar score. However, the other three models consistently improved their score compared to the GoPhone use-case. While the metro use-case in itself is more complex, the domain size (amount of named parameters) is lower, which may be the reason for the higher performance. Overall, the results again indicate a reasonable performance for the metro Wagon use-case.

6. Conclusion

This paper presented a configuration copilot that enables non-expert users to configure a product in natural language. The cooperative neuro-symbolic approach combines an LLM with a constraint solver to reliably support a product configuration. An early evaluation on the two use-cases of configuring the GoPhone feature model and a metro Wagon indicated practical feasibility. We believe that a configuration copilot is a valuable extension to GUI-based product configurators. For a productive implementation, limitations and future work mentioned in Sections 6.1 and 6.2 should be addressed.

6.1. Limitations

A limitation of our work is the size of the use-cases. Compared to real-world scenarios, the evaluated GoPhone feature model and metro Wagon are smaller and less complex. Additionally, the evaluation was done on a limited manually created dataset with 30 instances per use-case. While the most critical aspect of the architecture, the Formalizer, was evaluated, a formal evaluation of the Interpreter and the full configuration pipeline were omitted for the reason that a user study is required to evaluate these aspects. This paper demonstrates that creating a productive configuration copilot is feasible but does not study the extent to which value is provided to real users in a real-world scenario.

6.2. Future Work

To address the limitations of this paper, the configuration copilot shall be evaluated on more complex use-case from practice in a user study. The configuration copilot itself shall be extended: When a configuration as specified by the user is unsatisfiable, the configuration copilot shall suggest alternatives instead of reverting to the last satisfiable configuration. Additionally, soft constraints in the form of 'If possible, I would like to ...' shall be introduced.

References

[1] L. Zhang, Product configuration: A review of the state-of-the-art and future research, International Journal of Production Research 52 (2014) 6381–6398. doi:10.1080/00207543.2014.942012.

[2] M. Yi, Z. Huang, Y. Yu, Creating a sustainable e-commerce environment: The impact of product configurator interaction design on consumer personalized customization experience, Sustainability 14 (2022). URL: https://www.mdpi.com/2071-1050/14/23/15903. doi:10.3390/su142315903.

[3] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, C. Xiong, Codegen: An open large language model for code with multi-turn program synthesis, in: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[5] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, ACM Computing Surveys (2023).

[6] C. Ling, X. Zhao, J. Lu, C. Deng, C. Zheng, J. Wang, T. Chowdhury, Y. Li, H. Cui, X. Zhang, T. Zhao, A. Panalkar, W. Cheng, H. Wang, Y. Liu, Z. Chen, H. Chen, C. White, Q. Gu, C. Yang, L. Zhao, Beyond one-model-fits-all: A survey of domain specialization for large language models, CoRR abs/2305.18703 (2023). URL: https://doi.org/10.48550/arXiv.2305.18703. doi:10.48550/ARXIV.2305.18703. arXiv:2305.18703.

[7] D. Benavides, A. Felfernig, J. A. Galindo, F. Reinfrank, Automated analysis in feature modelling and product configuration, in: Safe and Secure Software Reuse: 13th International Conference on Software Reuse, ICSR 2013, Pisa, June 18-20. Proceedings 13, Springer, 2013, pp. 160–175.

[8] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, G. Tack, MiniZinc: Towards a standard CP modelling language, in: CP, volume 4741 of *LNCS*, Springer, 2007, pp. 529–543.

[9] A. A. Falkner, A. Haselböck, G. Krames, G. Schenner, R. Taupe, Constraint solver requirements for interactive configuration, in: L. Hotz, M. Aldanondo, T. Krebs (Eds.), Proceedings of the 21st Configuration Workshop, Hamburg, Germany, September 19-20, 2019, volume 2467 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 65–72. URL: http://ceur-ws.org/Vol-2467/paper-12.pdf.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[11] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2024.

[12] B. Wang, Z. Wang, X. Wang, Y. Cao, R. A. Saurous, Y. Kim, Grammar prompting for domain-specific language generation with large language models, in: A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 65030–65055. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/cd40d0d65bfebb894ccc9ea822b47fa8-Paper-Conference.pdf.

[13] G. Poesia, A. Polozov, V. Le, A. Tiwari, G. Soares, C. Meek, S. Gulwani, Synchromesh: Reliable code generation from pre-trained language models, in: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022, OpenReview.net, 2022.

[14] L. Pan, A. Albalak, X. Wang, W. Wang, Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 3806–3824. URL: https://aclanthology.org/2023.findings-emnlp.248. doi:10.18653/v1/2023.findings-emnlp.248.

[15] P. Kogler, A. Falkner, S. Sperl, Reliable generation of formal specifications using large language models, in: SE 2024 - Companion, Gesellschaft für Informatik e.V., 2024, pp. 141–153. doi:10.18420/sw2024-ws_10.

[16] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. C. Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, G. Synnaeve, M. Ai, Code llama: Open foundation models for code, 2023. URL: https://github.com/facebookresearch/codellama.

[17] MistralAI, Codestral introduction (2024). URL: https://mistral.ai/news/codestral/.

[18] AI@Meta, Llama 3 model card (2024). URL:

https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[19] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, 2023. arXiv:2310.06825.

[20] M. Mendonca, M. Branco, D. Cowan, S.p.l.o.t. - software product lines online tools, In Companion to the 24th ACM SIGPLAN International Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA (2009) 761–762. doi:10.1145/1639950.1640002.

[21] Gecode Team, Gecode: Generic constraint development environment, 2006. Available from http://www.gecode.org.

[22] M. LEVANDOWSKY, D. WINTER, Distance between sets, Nature 234 (1971) 34–35. URL: https://doi.org/10.1038/234034a0. doi:10.1038/234034a0.

[23] A. Falkner, A. Haselböck, G. Krames, G. Schenner, H. Schreiner, R. Comploi-Taupe, Solver requirements for interactive configuration, JOURNAL OF UNIVERSAL COMPUTER SCIENCE 26 (2020) 343–. doi:10.3897/jucs.2020.019.

[24] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, M. Stumptner, Configuring large systems using generative constraint satisfaction, IEEE Intelligent Systems 13 (1998) 59–68. URL: https://doi.org/10.1109/5254.708434. doi:10.1109/5254.708434.

[25] K. Czarnecki, S. Helsen, U. W. Eisenecker, Formalizing cardinality-based feature models and their specialization, Software Process: Improvement and Practice 10 (2005) 7–29. URL: https://doi.org/10.1002/spip.213. doi:10.1002/spip.213.