Available online at www.eccomasproceedia.org
Eccomas Proceedia UNCECOMP (2019) 509-533

ECCOMAS Proceedia

UNCECOMP 2019
3rd ECCOMAS Thematic Conference on
Uncertainty Quantification in Computational Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou (eds.)
Crete, Greece, 24-26 June 2019

RECOMMENDER TECHNIQUES FOR SOFTWARE WITH RESULT VERIFICATION

Ekaterina Auer[1] **and Wolfram Luther**[2]

[1]University of Applied Sciences Wismar,
D-23966 Wismar, Germany
e-mail: ekaterina.auer@hs-wismar.de

[2] University of Duisburg-Essen
D-47048 Duisburg, Germany
e-mail: luther@inf.uni-due.de

Abstract. *Methods with result verification such as interval analysis or affine arithmetic have been used successfully at least since the 1970s not only for dealing with the automated proofs that simulation results obtained using computers are correct, but also for taking into account the influence of bounded uncertainty in the input on the outcome of a simulation. There are many packages developed for providing basic arithmetic computations on different platforms, for example, filib++ in C++*[1]*, PyInterval in Python*[2]*, or such exotic implementations as JuliaIntervals in Julia programming language*[3]*, and this is only a small choice of tools for interval arithmetics. Moreover, there are packages for higher level algorithms such as solving initial value problems for ordinary differential equations (e.g., VNODE-LP), global optimization or linear/nonlinear systems of equations (e.g., inside C-XSC Toolbox). However, despite this abundance of software solutions, set-based methods with result verification are rarely used by an ordinary engineer for dealing with bounded uncertainty. In our opinion, one reason for this unpopularity is that engineers do not have time to compare the existing tools and choose the package that is most suitable for their task. To address this problem, we suggest using automatic recommendations.*

In this paper, we focus on software for solving initial value problems since it is important in many application areas such as biomechanics or automatic control. We show how modern concepts from the area of recommender systems can be employed to obtain an automatic suggestion about what tool to use for a given application and what prerequisites are necessary to be able to do so. We discuss in general what kind of data, metadata, quality criteria, metrics, and visualizations are required to be able to compare and recommend software with result verification. Finally, we present algorithms for recommendation and illustrate their functionality.

Keywords: software, scientific recommender systems, initial value problem, ordinary differential equations.

[1]`www2.math.uni-wuppertal.de/~xsc/software/filib.html`
[2]`pypi.org/project/pyinterval`
[3]`github.com/JuliaIntervals`

ISSN:2623-3339
Peer-review under responsibility of the organizing committee of UNCECOMP 2019.
doi: 10.7712/120219.6356.18823

1 INTRODUCTION

Digital assistance becomes more and more ubiquitous in everyday private and working life. Therefore, topics with applications to ambient intelligence and smart environments are of particular interest in the field of information science. Research trends focus on cross-cutting issues such as standardization, verification and validation assessment, or development of formal testing and quality criteria concerning, for example, reliability, performance and user satisfaction, which makes devising versatile metrics and agreeing on unified vocabularies across disciplines especially important. As demonstrated in [8], new software engineering priorities are in such technologies as search engines, recommender systems, and general data mining techniques.

The goal of recommender systems (RS) [2] is to aid consumers while they select from a variety of products, mostly with the aim to increase product sales (e.g., for online shops) or otherwise promote a given business (e.g., for social networks). Depending on the intended application, a number of other – operational or technical – goals can be set concerning, for example, relevance, novelty or diversity of recommendations. For generating suggestions, it is assumed that meaningful rules can be discerned about the way the consumers (users) select the products (items) or, vice versa, about the products most suitable for (a group of) customers. To achieve the mentioned goals, RS might take into account such characteristics as the intended product use, customer behavior, or product ratings and try to compile a ranked list from the multitude of offers (or predict a user's rating for a given item) according to criteria specified beforehand. Beyond the classical application in the area of e-commerce, RS are also in high demand for digital products, for media (including such different aspects as movies, news, or scientific publications), for healthcare, learning, or artificial-intelligence-based services. This new perspective requires novel, formally described standards, quality criteria and metrics [66] to determine the ranking in the sense of a multi-criteria optimization supported, if possible, by test environments (e.g., for software tools) and evaluation guidelines for recommendations.

Whereas evaluation of recommendations is a fairly well-known topic (cf. [2], Chapter 7), the related notions of testing and test environments might require a further explanation. Their obvious goal is direct certification of a RS, as, for example, in case of the data set and precise test descriptions provided in the context of the Netflix prize [4]. On the other hand, their development might be motivated by human factors: While some consumer groups might make decisions taking into account just the quality criteria, fulfillment of requirements, known benchmarks results, or recommendations from peers, other consumer groups, in particular, professionals, might want to parameterize and try out different digital products in a common test environment themselves to find the right product or the right solution in a given situation and for a specified task. A new research direction is therefore scientific RS (SRS) for digital products supported by test environments.

In our view, a scientific RS should possess the following characteristics:

1. It is a recommender for scientists providing a ranked list of items (e.g., software tools) suitable for the user's task.
2. It is usually knowledge based, context aware and multi-criteria; its item ratings evolve with time (e.g., requirements for software quality, its usability and the degree of interaction might evolve with time).
3. It provides recommendations which are relevant for a given user with respect to predefined (or inferred) criteria.

[4] www.netflixprize.com

4. It is flexible in the sense that (groups of) items or users with new attributes can be added easily to the set-up.
5. It is reliable, for example, is based on formally described criteria, uses reliable algorithms, provides explanations of recommendations for credibility and traceability of decisions, and takes care of good provenance of data (e.g., through common test conditions). Trustworthiness is supported through recommendation rating by users.
6. It can be supported by a test environment, which can take care of the cold-start problem and active learning.

By contrast, such RS subgoals as novelty, surprise, or diversity (cf. [2], page 3) need not be considered. Although context-aware, SRS does not necessarily need to take into account such domain-specific challenges as location and the social component. Social media for scientists such as `www.academia.edu` or `www.researchgate.net` belong to the general class of social RS such as `www.facebook.com` and might contain subcomponents which can be classified as SRS (e.g., recommending project partners).

In this paper, we apply the general principles and techniques of (scientific) recommendation in a specific context of verified software. After giving a brief overview of the state of the art in SRS along with suggestions of how it can be employed for our purposes (Section 2), we illustrate the concepts summarily in Section 2.2 using the application area of visual analytics, where automatic choosing of the appropriate visualization technique is a common software feature (often without the name of recommendation). Our focus is on recommending techniques for verified solution of initial value problems (IVPs) for ordinary differential equations (ODEs), an area where we would like to offer users similar options. In Section 3 we point out what is necessary to recommend a scientific tool in general. In Section 4, we describe first the available verified IVP solvers along with the necessary conditions and rules to be able to give a recommendation. Moreover, we provide and illustrate a specific algorithm using a small data set. Finally, we point out possible improvements and further research directions. Conclusions are in the last section.

2 SCIENTIFIC RECOMMENDER SYSTEMS

The purpose of scientific RS is to recommend an item (a ranked list of k items) to a scientist in a broad sense (e.g., engineer, researcher, teacher, student). This item can be a digital product, for example, a software tool or a technique, from various application areas such as visualization or education.

One classical formulation of a recommender problem is to determine a list of recommendations (for items) based on preferences and needs of a user (group), which we see as the most relevant formulation for a scientific recommender. Three general RS techniques are collaborative filtering (using information about user-item interaction), content-based methods (using information about attributes of users and items), and knowledge-based methods (using explicit information about user requirements). SRS recommend items for scientific tasks that fulfill predefined requirements and constraints and make use of all of the mentioned general techniques where necessary. To establish a list of recommendations, SRS often rely on object or case-based approaches accompanied by filtering and learning algorithms, similarity measures to compare items (cases), quality criteria and metrics to select and rate suggestions, and intelligent algorithms to match users' quality criteria with product properties/descriptions or to find people with similar interests/profiles and expectations for the product.

In this section, we identify the most common SRS topics first. After that we describe the

subtopic of visual analytics in more detail. Note that our literature review is not supposed to be complete and has a narrow focus. For general overviews of RS and their research topics, see, for example, [2, 32], and for SRS [22, 25, 53, 58, 59, 49], which are mainly about RS in education and paper recommending. In [50, 14, 26], the state of the art is described for aspect-based RS which make personalized recommendations taking into account the users' opinions about aspects of the rated items extracted from their reviews.

2.1 Topics for scientific recommendation

Below, we identify research topics in scientific recommendation (in capital letters) and exemplify them by (a) relevant publication(s). The considered SRS mainly use a case database with benchmarks for problems and their solutions; retrieve similar tasks; reuse, adapt and revise solutions and retain new cases and new user preferences. That is, they can be classified as mostly knowledge-based and employ appropriate quality criteria and metrics. Case-based and feature-based SRS continue to evolve by including new user groups, their tasks and environments as shown in [39].

SRS-SE recommend relevant activities for software engineering (SE) tasks and support developers during programming of software components by providing "information items estimated to be valuable for a software engineering task in a given context" [56]. Specifically, SRS-RE offer help in the area of requirements engineering (RE) considered to be "one of the most critical places in software development" [22] by employing the whole range of recommender techniques from collaborative filtering to social media related algorithms.

SRS-P make a choice of relevant papers (P) from a specified scientific field. Scientific paper recommender systems are extensively described in [59], supplemented by several new metrics and a comparative/contrasting definition of various recommendation tasks. An important subtask is to extract semantic relations between keywords from scientific articles in order to support users in the process of browsing and searching for content in a meaningful way [38]. SRS-E help students and teachers to make choices (e.g., of suitable courses) in the educational (E) context [23]. Further tasks for SRS-E are given in [4], where the authors describe an RS that can be applied for finding experts in academia, for example, supervisors for students' qualifications or research, reviewers for conferences, journal or project submissions, or partners for R&D proposals. SRS-STI, that is, RS for scientific and technical information (STI), are addressed in [48]. Here, a more general point of view is adopted by combining the angles of SRS-P (e.g., scientific libraries), SRS-E (e.g., e-learning) and others. The privacy issues and the cold start problem are addressed and several algorithms for the generation of behavior-based recommendations are explored there.

The next topic is the one of the most relevant w.r.t. the goal of this paper. SRS-PSE provides recommendation for problem-solving environments (PSE) [28, 27, 67]. In [67], the project CB-Matrix is described – an early development in the area of devising "intelligent recommender components" to assist scientists in choosing and applying scientific tools. The developers of the project PYTHIA [28, 27] start out by recommending software/methods for partial differential equations and then extend their methodology to enable users to prototype their own recommenders on the basis of their own databases and specifications for interaction with underlying execution environments. The resulting customizable web-based platform MyPYTHIA does not seem to be freely accessible online anymore[5]. MyPYTHIA leaves the problem of a

[5]The service swMATH `http://swmath.org`, which is a SRS-P itself and provides information on mathematical software based on the analysis of publications [9], only supplies links to papers for the keyword

common test environment out of consideration [27]. SRS-R (SRS for reliable (R) or verified software/hardware) and SRS-VA (SRS for visual analytics, VA), which are in the focus of this paper, can be considered as subtopics within this general setting. Even if it is not explicitly termed as a RS in such publications as [7], the mechanism behind choosing visualization techniques based on optimization of a metric w.r.t. quality criteria can be seen as such. More information about this topic is in Section 2.2. SRS-R deal with reliable hardware and software components, use reliable algorithms and include evaluation strategies for the system outcome, even if ground truth to assess accuracy is missing. This topic is described in detail in Section 4. Finally, SRS-AS, RS for assistive software (AS) [24], enable existing interoperability architectures to automatically select the most suitable assistive software for a given interaction with a specific electronic target device taking into account the user's benefit and disabilities.

Knowledge-based RS often employ ontologies (constructed beforehand in a 'intelligence' step, e.g., from user reviews) for generating recommendations. Ontologies provide a structured framework for modeling concepts and relationships between scientific domains of expertise. They are a prerequisite for development of domain knowledge metadata bases for modeling, communicating and sharing knowledge among people (or problem-solving applications). A lot of work has been done in this field, also from the angle of artificial intelligence. For example, PROTEGE-II [62] is an implementation of a methodology for building knowledge-based and domain specific knowledge acquisition systems. The tool provides protocol-based decision support in a specific medical domain. Another tool, CEDAR OnDemand [11] allows users to enter ontology-based metadata conveniently through existing web forms from their own repositories. The web page contents is analyzed to identify the text input fields and associate them with automatically recommended ontologies. Finally, there are modeling languages based on ontologies. For example, the publication [17] shows how to employ the unified problem-solving method development language (UPML) as a comprehensive framework for modeling libraries of methods. UPML provides a hierarchy of concepts to specify knowledge components. In particular, the description of a method includes a competence (defined by a set of input and output role descriptions as well as preconditions and postconditions, e.g., formulas for inputs and outputs), a separate method ontology (definitions of the concepts and relationships of a method) and its associated operational description.

2.2 Recommending a visualization

A publication on scientific recommenders would not be complete without mentioning the field of visualization, which is a very extensive topic. A lot of work has been done on choosing the right visualization for the problem at hand, often without explicitly calling it a recommendation. In this section, we mention the most important publications in this area from our point of view and describe an application of such techniques in the field of steel production.

2.2.1 Short overview of recommender tools in visualization

The general goal of SRS-VA is to automatically suggest a visualization providing insight about the data under consideration, ideally taking into account their characteristics and domain as well as individual user preferences. Accordingly, approaches to visualization recommendation can be classified loosely into four categories [34]: RS based on data characteristics, RS (additionally) using representational goals, RS employing domain knowledge to improve recommendations, and RS relying on explicit interaction with users to infer their preferences. The first group can be considered as the most widely spread one since it was explored long before

the term RS had been applied to VA. The others appeared as a result of cross-cutting research in such areas as RS, data science, information visualization, and artificial intelligence. The boundaries between groups are not sharp so that there are methods using (parts of) techniques from the other groups. Below, we exemplify the concepts with appropriate RS references. For more information, see [34].

In the first group, the authors of [40, 60] suggest encoding ordered sets of user-specified data and metadata descriptors by visual variables (e.g., size, texture, color, shape). They develop a compositional algebra to enumerate the space of encodings and apply a set of visual integrity criteria to prune and rank the set of visualizations. This approach resulted in algebraic specification language VizQL with the help of which both the structure of a view and the complying queries can be specified and used to fill the structure with data. Moreover, the module Show Me [40] introduces a set of heuristics to extend automatic presentation to the generation of tables of views (small multiple displays) and recommend chart types. This research is implemented in a commercial tool Tableau[6]. An example of a free tool from the same class is the web application Voyager [68, 69]. The Voyager approach uses statistical and perceptual measures for finding out interesting relationships between data and transformations and allows for automatic generation and interactive steering of views as well as refinement of multiple recommendations.

One of the research goals in the second class of SRS-VA is automating generation of user tasks from natural language descriptions instead of creating them manually [34]. In the latter case, there is a connection with formal modeling methods for user interfaces/interaction. In the former case, advanced linguistic techniques are necessary. An example here is the tool Improvise[7]. The tool SemViz[8] [46] belongs to the third group and uses knowledge ontologies from the semantic web for adaptive semantics visualization. Similarly, a knowledge base of various ontologies is used in [65] to recommend visualizations. Here, the whole range of techniques from the previous classes is employed: Although rule-based and functional requirements govern discovering and ranking of potential mappings, such factors as device characteristics, data properties and descriptions of tasks influence the pre-selection and the final ranking. Finally, tools like VizRec [45] or VizDeck [35] belong to the last class and employ information about perceptual guidelines and explicit feedback about user preferences.

The publication [7] gives a comprehensive overview of metrics to compute the quality of a visualization which have been introduced and discussed for different information visualization techniques in recent years. The *quality-metric-driven automation* layer added to the visual analytics pipeline can serve directly as the basis for making data characteristic oriented recommendations, which is (implicitly) suggested to be done by multi-criteria optimization. In particular, the authors cover node-link diagrams and matrix representations for relational data; parallel coordinates and pixel-based techniques for multi-dimensional data; scatter plots and scatter matrices for high-dimensional data; TreeMaps for hierarchical data; radial visualizations when focusing on one dimension (e.g., a person); glyphs, line and bar charts for uncertainty visualization; and, finally, typographic visualizations and tag clouds for visual representation of text data. Additionally, geo-spatial data visualizations are examined separately as a case of special purpose visualization. As the authors explain, the selection focuses on fields in which quality criteria and quality metrics along with their underlying concepts, tasks and evaluation efforts are (semi-)formally described and can be examined analytically. Moreover, they present a high-level overview of visual exploration goals supported by the majority of metrics, for ex-

[6] www.tableau.com
[7] www.cs.ou.edu/~weaver/improvise
[8] knoesis.org/semviz

ample, clutter reduction filtering out noisy views, identifying data groups and partition clusters, establishing relations between dimensions, filtering out outliers, or preserving original data properties in the mapping process while reducing the number of dimensions. However, no user studies are conducted that compare the different metrics for different tasks and different data characteristics from a human-centered point of view.

2.2.2 Challenges in visualization recommender research

The directions of research in SRS-VA are aimed towards stronger involvement of human factors (e.g., higher interactivity) and domain specifics in generation of recommendations which might necessitate higher use of formal languages, standards or ontologies (e.g., for encoding tool and task categories). Filters concerning user experience and further (better) quality criteria and metrics to rank recommendations remain topics of interest [57]. A further challenge is that there exist many tools and methodologies for visualization, which requires possibly expensive filtering, which in turn influences the efficiency. Besides, finding competent users is necessary for selecting the right quality parameters out of a large number and for specifying optimization goals. Finally, although extensive research in this direction exists [7], it is a challenge to devise computable quality measures for optimization. For that, representative situations and datasets, users, tasks, and quality criteria are necessary. Quality measures could be derived from evaluation studies concerning task categories, user experience and interaction styles; concerning visualization tools (with the focus on performance, accuracy, usability, result presentation readability, integrity); and concerning data and metadata quality.

2.2.3 An application to steel production

In the previous subsections, we described recommendation methods relying heavily on formalizing different concepts in visualization. It is also possible to approach the task empirically, which overlaps somewhat with the class of RS based on explicit user interaction. For example, users are urged to explore a small number of parameter variants using large singles and small multiples as alternative views in [21], allowing for efficient data analysis.

A similar approach is adopted by the inclusion processing framework viewer IPFViewer 2.0, developed under guidance of the second author and employed in the area of steel production for analyzing (big) data collected about non-metallic inclusions and other defects in steel samples [61]. Extensive interviews were conducted with experts after the initial version 1.0 had appeared, during which alternative visualization concepts (i.e., various forms of multiple views) had been shown to users. IPFViewer was adapted to the outcome of the survey in its version 2.0, which in turn was evaluated again by the same experts. That is, a number of visualization recommendations (for highly specialized experts) were generated and evaluated comparatively through user feedback w.r.t. their suitability.

IPFViewer 2.0 takes into account process parameters such as intentional settings or measurements taken during monitoring of various steel grades and their metadata, defect parameters, descriptors and volume data for each defect, isoperimetric shape factors (e.g., volume or surface area), sample parameters (e.g., milling machine slices of the steel surface), and statistical descriptors of the defects (e.g., the sample cleanliness). It performs 3D reconstruction of cracks, non-metallic inclusions or pores. The tool can analyze the ensemble data set in various ways, for example, detect outliers to identify samples that differ from the others by position, size, type and number. To rate steel quality, it carries out trend analysis to study the influence of different

process parameters on the steel samples and variance analysis to examine natural fluctuations within the samples and desired variations that result from process parameters. When required, IPFViewer relies on incremental, approximate analysis techniques to ensure the responsiveness of the application while sufficient precision is guaranteed for queries with fast response times.

The steel production facility workers are now able to quickly and interactively analyze data with millions of data rows. The resulting data tree is visualized as a huge grid in a scrollable area. Each grid cell incorporates a multiple view system with such standard visualization techniques as scatter plots, bar charts and trend graphs. Steel experts examine the histogram about defect diameter and the largest found defects to evaluate a sample quickly without having to analyze each defect manually. They can also scroll through all the samples and compare them, create and save various layouts that visualize different aspects of the data in order to confirm or refute hypotheses.

3 WHAT IS NECESSARY TO RECOMMEND A SCIENTIFIC TOOL?

In Figure 1, three general steps most SRS have to undergo to generate a recommendation are shown. The purpose of the first one is to extract information which describes a given user's request for a recommendation. This retrieval can be automatic (e.g., identifying keywords from texts), interaction-based (e.g., asking users to enter keywords) or manual (e.g., rigidly fixing the keywords). Here, the base data/metadata set is produced. In the next step, new information is generated, for example, taking into account similarity measures or based on ontologies, possibly including machine learning algorithms. This produces candidates for recommendation. Finally, the candidates are ranked according to predefined criteria or metrics and the resulting list of recommendations is conveyed back to the users. An important additional step is evaluation of the produced recommendations. For example, if there is a common testing environment for the items of interest, the recommendation can be validated additionally and the feedback about these validation results reused at the intelligence step. There are also other possibilities for evaluation such as user studies, see [2], Chapter 7.

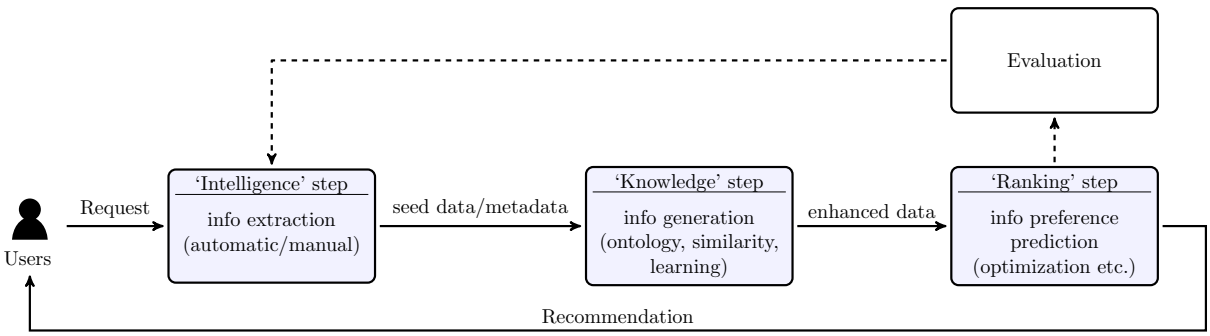

Figure 1: General stages in RS. Steps strictly belonging to a RS are shown in blue.

To illustrate these steps and to see what is necessary to implement a scientific recommender, let us consider a relatively simple example from [67]. The task of the RS from [67] is to recommend data structures (e.g., block matrices) for solving large sparse linear systems based on previously solved use cases. At the first step, features of matrices are determined (e.g., number of non-zeros, degree of bandedness) and a database of past information on pairs "matrix"-"data structure" is created. Possibly, the data have to be normalized beforehand. At the next step, a predefined similarity metric is used to be able to determine matrices similar to a given new

one. To improve similarity detection, tests based on genetic algorithms can be carried out here to automatically determine a 'good' set of feature weights. At the third step, the most suitable data structures for these similar matrices are determined and ranked for the recommendation according to a 'measure of goodness' (i.e., performance in flops). The recommendations are evaluated in cross-validation tests. As shown in [27], such principles can be generalized and used to generate recommenders themselves.

Since the goal of a SRS is to provide a ranked list of tools best suited for a given user task as explained earlier, we describe a possible approach to choosing the most suitable item (or a list of them) and relate it to the scientific tool context. An interesting ranking method based on keywords is introduced in [51, 52]. A more sophisticated ranking algorithm is described in [70]. For textbook approaches in network context, see [2], Chapter 10.

As already explained in relation to VA, *multi-objective optimization* can play an important role as an RS technique and be seen independently of visualization context. A quality function q and the associated algorithm A are defined in dependence on the problem-solving tool v, several descriptors (e.g., those pertaining to data, users, and tasks) and some side conditions. As proposed in [7], the algorithm A has to solve a multi-objective optimization problem in order to find a problem-solving instance v (or a ranked list L of such tools v) to maximize (minimize) q. The choice of parameters to optimize and their ranges depends on the task definition, requirements, equality and inequality constraints, valid standards or measurements/experiments for validation. This choice can be made by trial and error, searching or filtering. The function q is characterized by quality criteria and quality metrics defined in the context of the user group and its profile, the task and its model, the data, metadata and data types mapped to the tool v belonging to a predefined set of computer-based problem solvers, the hardware and its interfaces. The quality criteria encompass performance of the task completion including effectiveness and efficiency, reliability criteria for the input data that need to be mapped by v to the outcome space accurately and efficiently. If the optimization problem can be solved automatically, then the problem solving tool v or tool selection and its/their quality metric parameters fulfill the requirements and side conditions and can be recommended to the user (group) for drawing conclusions and making decisions. Typically, requirements concern the solution or its enclosure under uncertainty in parameters. The success of this approach depends on whether an effective and efficient implementation A of the target function q and its computation can be provided.

In order to produce L, a strategy similar to [59] can be employed. There, a similarity-based diversity metric m_{div} is considered for a set P of scientific papers p_i as a normalized sum

$$1-m_{\text{div}} := c^{-1}\cdot\sum_{i\neq j} m_{\text{sim}}(p_i,p_j) \in [0,1], \text{ where } c = (|P|(|P|-1))\cdot\max_{i\neq j}\ m_{\text{sim}}(p_i,p_j)\ , \quad (1)$$

which can be adapted to scientific tools. That is, such tools can be seen as similar if their values for a given quality measure differ only slightly on a set B of benchmark problems. Further, the term 'coverage' is introduced in [59] as "the extent to which all important aspects and subtopics of a scientific field are covered by a set of papers". It is an open research topic to develop a method to similarly cover a given problem space with a small number of scientific tools that solve the benchmark problems w.r.t. given requirements, constraints and quality criteria, for example, performance, accuracy, and usability. That means describing the problem space as a multidimensional space T depending on problem descriptors and compiling a small number of tools in a list L that clusters the space T. This can be done by using an average similarity

distance

$$Av(d_{\text{sim}}(L)) := \frac{\sum_{i \neq j} d_{\text{sim}}(v_i, v_j)}{|L|(|L|-1)} \tag{2}$$

for a set $L \subseteq T$ of scientific tools v_i and the similarity distance d_{sim} by constructing a sequence $Av(d_{\text{sim}}(L_i))$ with further tools $v_{i+1} \in T$ as far as possible from the already taken item set $\{v_1, v_2, \ldots, v_i\} \subseteq T$. For this, we arrange the distances according to their value in descending order and start with two most distant items v_1, v_2. The process can be terminated if the last distance $d(L_i, v_{i+1})$ falls below a certain limit c. Then for each $v \in T$ there is at least one item $v_i \in L$ with $d_{\text{sim}}(v_i, v) = c$.
To summarize, what is needed for SRS is the following.

Database User features, item features, user-item information
Information generation strategies Classification, ontology
Metrics Means for establishing similarity and goodness (quality criteria, weights to reflect situational context)
Ranking algorithms based on metrics such as the similarity-based diversity metric (1) or the the cosine metric (6)
Common test environment (optional): Database generation, evaluation.
Generalization (optional): Means to decouple a recommender from the actual feature vector or metric/criteria instantiation

4 A SCIENTIFIC RECOMMENDER FOR IVP SOLVERS

In this section, we describe an algorithm to recommend verified initial value problem solvers (IVPS) for ODEs and its possible generalizations. Verified methods [44] are constructed in such a way as to provide a mathematical guarantee that a solution obtained on a computer is correct. IVPS generate numerical sets that are mathematically proved to contain exact solutions. They are useful in different contexts, for example, for computer-assisted proofs [63, 64] or for propagating bounded uncertainty through systems [54]. There are many free libraries implementing verified IVPS techniques, which we describe in some more detail in Subsection 4.1. However, an average engineer is disinclined to use them, main reason being the difficulty to choose the right method for a given problem without having the full knowledge about the subject. Some verified methods might be too simple to be used for an advanced application leading to very conservative or pessimistic results; other methods might be too prohibitive computationally. This led us to the idea of implementing a common web-based environment for testing such verified IVPS, which can serve as a basis for a recommender [6].

As far as we know, there are no comparable recommenders for non-verified, normal floating-point arithmetic based IVPS. Such recommendation portals as MyPYTHIA [27] could probably have been used for that purpose but does not seem to be online anymore. Note that the MyPYTHIA application to partial differential equations described in [27] could serve as a good basis for the appropriate IVPS recommender. There are several web platforms for gathering benchmarks, testing and comparing traditional non-verified IVPS, most notably TEST SET at `pitagora.dm.uniba.it/~testset/` building on research from DETEST [29] and similar. Besides, the service swMATH [16] semi-automatically manages the existing Web information about mathematical software. However, verified IVPS have to be compared based on different criteria: for example, they always produce “correct” solutions, that is, the reliability of the results does not need to be tested. After describing available software for verified solution of IVPs and our testing environment VERICOMP, we analyze existing literature concerned with

testing and comparing them to identify possible quality criteria and problem classification in Subsection 4.2. The rest of this section is devoted to our recommender algorithm (including an illustration) and possible improvements.

4.1 Verified IVP solvers and VERICOMP

A number of most widely known IVPS are summarized in Table 1 along with some of the newer tools. Some of these IVPS are also suitable for computing solutions for hybrid system dynamics (e.g., Flow*), algebraic-differential equations (CORA) or Poincaré maps (Isabelle), some of the tools additionally provide non-verified solutions (e.g., CAPD) or the use of multiple-precision arithmetic (e.g., kv).

It can be seen from the table that the IVPS are based on very different algorithms (Column 4) with different data structures implemented in different programming languages (Column 2) using different verification concepts (Column 3). A further point is that their performance often depends on the right choice of their settings, which should be preferably tuned to the given problem by their respective developers (Column 5). For the sake of presentation clarity, we only show parameters which are important in our opinion. The time span for the simulation, that is, the initial and the final integration time, is also an important setting and can be specified within all IVPS. Although most of the solvers use only result verification, a lot of effort has been devoted to formal verification of solvers' codes [31, 41] recently. VNODE is a tool relying on the concept of literate programming [37, 47] for code verification. Literate programming allows a human expert to assess in a comfortable way if a code is correct. The list of solvers is not complete, for more software consult, for example, `cps-vo.org/group/ARCH/ToolPresentations`.

A forum for comparing software for verification of continuous and hybrid dynamical systems is offered by the workshop ARCH [1] and its friendly competition[9]. One of the aims is to establish a curated set of benchmarks submitted by academia and industry. This extensive information service gathers and makes accessible benchmark problems, tool presentations, and experience reports in form of papers. However, the approach has shortcomings. The workflow of the competition is to join a group first, then determine the set of problems from the ARCH pdf repository, perform the tests, and, finally, prepare a report. This workflow is not automatized, the responsibility for the correct implementation and testing with the benchmarks for given tools lies with the user/developer, there is no common testing environment, and the results from different reports are not immediately reusable since stored in papers and not in any kind of a common database precluding an automatized recommendation. The long-term goal of our web-based platform VERICOMP [6] is to provide such a common, automatized, recommender-enhanced comparison environment.

VERICOMP is a service for actually comparing verified initial value problem solvers for systems of ordinary differential equations using common comparison conditions. One possibility to employ it is for developers of new IVPS in order to relate their tool to the state of the art. Another possibility is to employ it to decide what solver is the best for a given problem. For users be able to do so at a glance, VERICOMP uses work-precision diagrams, solution plots, and text tables. Here, developing further VA strategies for representing these heterogeneous, large data is our future work. The gathered information is stored in a MySQL database.

The tests with verified IVPS might take considerable time. Therefore, a recommendation should be provided based on similar problems from the database. The RS results can

[9] `cps-vo.org/group/ARCH/FriendlyCompetition`

be additionally validated by actually performing the available tests. Three solvers Vnode-LP, ValEncIA-IVP, and RiOT with various parameters were provided for testing in the old version of VERICOMP under `vericomp.inf.uni-due.de`, the service of which is unfortunately discontinued. VERICOMP 2.0[10] taking into account generalizing features from Section 4.4 is under construction. At the moment, only the feature of adding IVPs to the database and browsing them is accessible. We work on implementing the functionalities discussed in Sections 4.2,4.3.

It is our long-term goal to provide a common environment for testing all the verified solvers mentioned above, which means that a semi-automatic procedure for adding a new solver to VERICOMP is needed. However, this is extremely difficult due to differences in interfaces, concepts, programming languages and platforms. Therefore, we concentrate on the intermediate goal of providing a database which needs to be filled by IVPS' developers themselves. This leads to the lesser challenge of providing the set of problems in the form well suited for running tests with them with different software, which we work on at the moment. This requires a survey on opinions of expert users.

4.2 Existing comparisons and quality

Emerging and old verified software is permanently being tested and compared with some standard benchmarks. For example, VERICOMP's problem database was used as a benchmark for new IVPS in [18, 19]. More tests are described and made available through papers at ARCH. However, every paper presenting a new solver (e.g., [31], [41]) features tests and comparisons according to some criteria relevant for the authors and using benchmarks the authors consider appropriate. To be able to have an overview over the whole range of available possibilities, it is necessary to standardize the comparison criteria and the tests as well as devise various benchmark problem sets. In this subsection, we analyze the publications [12, 19, 20, 31, 33, 18] from Table 1 with the goal to establish a common set of quality criteria and testing aims. Besides, we name the problems most commonly used as benchmarks.

The *quality criteria* can concern performance, accuracy, efficiency, or usability. Usability is often of a minor significance in IVPS tests, which is not entirely justified since such factors as the ease of interfacing an IVPS with a given application might play a crucial role in practice. In verification context, accuracy means the degree of pessimism in the resulting enclosure (e.g., overestimation). Overestimation is not always easy to characterize: the width of the resulting enclosure is only a good indicator for that if all problem parameters are crisp and do not contain any uncertainty [6]. More research on overestimation characterization for tests is necessary. Statistics on the following quality criteria were actively gathered by the old version of VERICOMP:

C4 wall clock time (t_{c}) at a predefined integration time t_{out} (performance),
C5 user CPU time t_{us} w.r.t. overestimation e_{us} at t_{out} (efficiency), and
C6 time to break-down (t_{bd}, accuracy), possibly bounded from above by a certain limit t_{max}.

Here, e_{us} is mainly (but not always) assumed to be characterized by the resulting enclosure width [5]. These criteria allowed us to produce quite accurate recommendations using the algorithm from Section 4.3, see [5]. Besides, the following further criteria can make sense [30]:

C1 Number of arithmetic operations at a time step

[10] `vericomp.fiw.hs-wismar.de`

C2 Number of function/ Jacobian, etc./ inverse matrix evaluations
C3 Overhead (the overall user CPU time minus the user CPU time for function evaluations [30])
C7 Total number of steps and the number of accepted steps

Besides C4 and C6, the most widely used quality characteristics concerning performance, accuracy, and efficiency, resp., in publications from Table 1 are

C8, C9 User CPU time and the width of the enclosure at t_{out}
C10 CPU time to achieve a certain prescribed enclosure width over the time span $[0, t_{\text{out}}]$

As the eleventh criterion **C11**, usability based on empirical (online) studies should be introduced. The *testing aims* in the considered papers are to characterize a given solver w.r.t. the state of the art. However, users also like to find a good IVPS for their given application (cf. experience reports at ARCH). This goodness can be characterized by various scenarios, for example, an offline simulation with very high accuracy (e.g., for particle colliders) or fast online verified simulation over short time spans (e.g., for control).

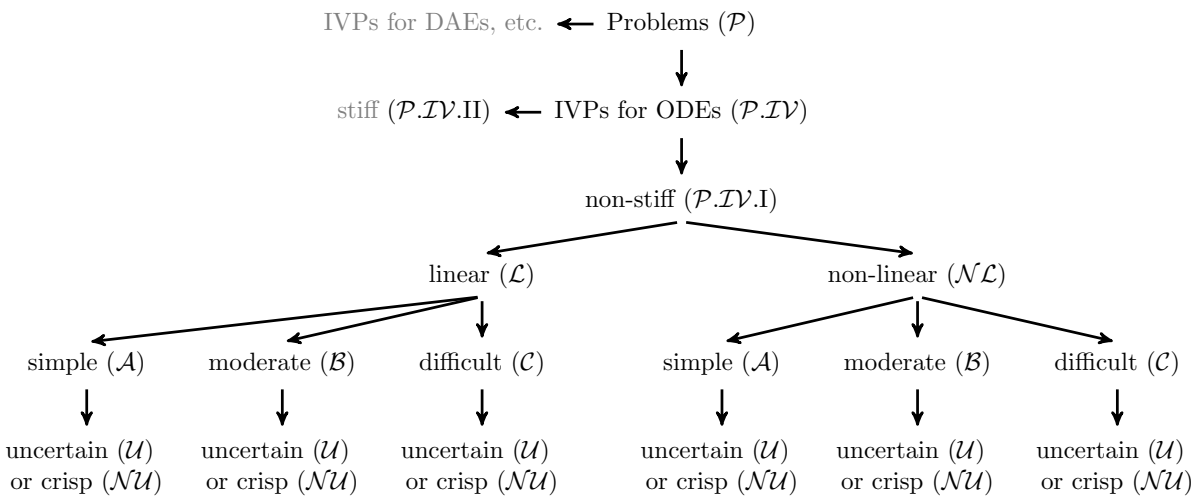

Figure 2: Classification of benchmark problems for verified IVPS [6].

The most common *benchmarks* used for solver characterization in the considered papers are the Lorenz system, the Rossler system, (Lotka-)Volterra equations, the oil reservoir problem, the harmonic oscillator as well as problems from the DETEST benchmarks. Several papers use old VERICOMP version benchmarks, one difficulty being that the problem IDs changed in the new version. Besides, the benchmark set consisting of over 73 problems needs structuring. For example, automatically extractable, distinct problem benchmark sets for asserting common ground (e.g., the five problems mentioned above), for advanced testing, or for practice-oriented testing can be devised. For more details about the form of the mentioned test problems, see VERICOMP under `vericomp.fiw.hs-wismar.de`. For the benchmarks, we suggest using the classification in Figure 2. The main classes of linear and non-linear problems have three subclasses: simple (possibly with exact expressions for solutions), moderate (w.r.t. their dimension, order, etc.) and complex or difficult problems. In each of these subclasses, we differentiate between problems with uncertain and crisp parameters. The use of this classification was justified in [5]. Further problem classes such as ODEs with delays or non-smooth right sides as well as hybrid systems or systems of differential-algebraic equations can be incorporated into it.

However, the classification needs to remain flexible to be able to reflect factors which interest users at a given moment, for example, chaoticity or cooperativity.

The considered papers mostly use tables and trajectory plots to present comparison results. Only [18] uses other visual aids (spider diagrams) for that purpose. Occasionally, just one set of IVPS parameters (cf. Table 1, Column 5), which additionally varies from example to example sometimes, is employed and the strategy behind the choice of settings is not always clear. A comparison using these settings consistently would be more interesting from the point of view of finding out and recommending such settings automatically.

4.3 A recommender for IVPS for ODEs in VERICOMP

We can think of the following tasks a recommender for IVPS might be required to solve:

1. Recommend a solver (a ranked list of solvers) for a new user problem under consideration of their specific tasks (online/offline simulation, etc.)
2. Find a coverage set L for a problem set B

In this subsection, we describe a formal basis for such a IVPS recommender. We plan to make it accessible in the near future.

Following [36], we define a SRS for IVPS as a 6-tuple $< U, T, L, K, P, S >$, where U represents the user, T is the entity set (of items) , $L \subseteq T$ is the set of recommended items, $K = K(P, T, S)$ is the context, P stands for the user profile and S for the situation. To produce a recommendation, we maximize a certain utility function χ depending on the user, the context, and the set of recommended items.

To solve the recommendation tasks mentioned above, we identify U with the problem a user wants to solve. Therefore, P coincides with the problem characteristics defined by the classification in Figure 2. T contains solvers characterized by their specific settings and S is described by the type of application the users have in mind for their problems (e.g., online/offline simulation). Note that the context K is independent of T, because the number of solvers does not change during a session. The utility function can be a weighted sum of normalized values for each criterion **C1**,. . .,**C11**:

$$\chi(v, u) = \sum_{i=1}^{m} \omega_i n(C_i(v, u)), \quad v \in T, \quad u \in U, \quad \sum_{i=1}^{m} \omega_i = 1, \quad m = 11, \tag{3}$$

where ω_i are the weights, $n(\cdot)$ a normalizing function, and $C_i(v, u)$ a function returning the value of the criterion i for solver v and problem u, for example, as shown in Eqs. (5). Note that we assume that v is not merely one of the solvers, but rather a solver with certain settings (e.g., ValEncIA with the stepsizes of 0.025, 0.0025 is represented by two separate items in T). To produce a recommendation, we use the multiattribute utility collaborative filtering with the given criteria and weighting according to the situation $s \in S$ [43]. The first step in the process of filtering is to establish similarity to a (group of) problem(s) u from the database U with the help of a measure $\mu(u)$. In our case, we can define a simple $\mu(u) := \mu(l(u), c(u), f(u))$ as depending on the linearity $l : U \mapsto \{\mathcal{L}, \mathcal{NL}\}$, complexity $c : U \mapsto \{\mathcal{A}, \mathcal{B}, \mathcal{C}\}$, and the presence of uncertainty $f : U \mapsto \{\mathcal{U}, \mathcal{NU}\}$. It returns all problems from the class uniquely defined by the values $l(u)$, $c(u)$, $f(u)$:

$$\begin{aligned} \mu &: U \mapsto 2^U \\ \mu(u) &:= \{\tilde{u} \in U | l(\tilde{u}) = l(u), c(\tilde{u}) = c(u), f(\tilde{u}) = f(u)\} \ . \end{aligned}$$

The set $\mu(u)$ constitutes the user profile P. The next step is weighting: according to the situation s, a vector with weights $(\omega_1, \ldots, \omega_m)$ is determined by the function ν as

$$\begin{aligned} \nu &: S \mapsto [0,1]^m \\ \nu(s) &:= (\omega_1, \ldots, \omega_m) \ . \end{aligned}$$

The third step is finding the appropriate neighborhood for the problem u. In our case, this neighborhood coincides with P. In the final step of the recommending process, we rate the available solvers $v \in T$ with the help of the normalizing function

$$n : \mathbf{R}^+ \to [0,1], \quad n_{k_1,k_2}(x) = \frac{1}{1 + e^{1-(x-k_1)/k_2}}, \quad k_1, k_2 \geq 0 \ , \tag{4}$$

with k_1 and k_2 being real heuristic parameters depending on the data, and the function χ from Eq. (3). We use $k_1 = 40$ and $k_2 = 10$ in the following.

To see how the recommendation work, consider the following example. Let an engineer be interested in simulating a non-linear, simple problem with uncertainty in a verified way. Suppose the similarity measure μ returns a set consisting of two problems u_1 ($\dot{x} = -\frac{1}{2}x^3$, $x(0) = [0.5, 1.5]$) and u_2 ($\dot{x}_1 = 1$, $\dot{x}_2 = x_2 \cos(x_0)$, $x_1(0) = 0$, $x_2 = [0.9, 1.25]$) belonging to the class $\mathcal{P.IV.I.NL.AU}$. Suppose further that the data concerning the quality criteria **C4**, **C5**, and **C6** shown in Table 2 for three solvers RiOT, Valencia, and VNODE with three different settings are recorded for these problems in the database. Finally, suppose that the engineer's goal is to simulate the problem online over short time intervals, which defines the situation s. If we restrict ourselves to the three criteria for which the data are recorded, the wall clock time (**C4**) and the relation of user CPU time to overestimation (**C5**) are equally important, whereas the time to break-down and the width of the enclosure there (**C6**) do not play much of a role. Accordingly, the assigned weights for $i = 4, 5, 6$ are 0.4, 0.4, 0.2, resp., and zero otherwise. The neighborhood is the set $\{u_1, u_2\}$.

Now we are ready to rate the solvers. A pre-normalization requirement is that bigger criterion values should correspond to better performance. Therefore, they are computed as follows for a solver v and a problem u:

$$\begin{aligned} C_4(v,u) &= 1/t_{\text{c}} \\ C_5(v,u) &= 1/(e_{\text{us}} \cdot t_{\text{us}}), \\ C_6(v,u) &= t_{\text{bd}}/e_{\text{bd}} \ . \end{aligned} \tag{5}$$

The ratings obtained using these definitions in formula (3) are shown in the last column of Table 3 (rounded). A higher rating ($0 \leq \chi(v,u) \leq 1$) indicates better performance. Note that the problem was not actually simulated to make the recommendation. From Table 3, it is clear that the criterion values are always the highest for VNODE with the 15th order of Taylor expansion in the situation s, a recommendation in good accordance with the data in Table 2. The averaging order plays a role: The criterion values can be computed as given in Eqs. (5) for each problem and averaged. This value can be then used in the formula for the rating χ. Alternatively, ratings $\chi(v_i, u_i)$ can be computed for each benchmark u_i and averaged afterwards, which usually provides a better separation between the ratings (therefore, our formula for χ is given such that this dependence is recorded directly). The main difficulty is to find a good normalizing function for the broad range of criterion values (5). Users can validate the recommendation by running the standard test on the problem u. Note that recommendations depend greatly on the information

in the data base. Besides, they are produced for a *problem class* and not for a *particular problem*, making a flexible classification procedure a must. Our further work includes improving this recommender principle through the use of better metrics (e.g., (1),(6)), averaging (e.g., (2)) and normalizations.

4.4 Improvements

Recommendations depend significantly on problem and solver features. The problem features in VERICOMP are the right side of the IVP, initial conditions, the integration time interval, parameters, the exact solution (if available), the assignment to a problem class and a textual description concerning, for example, the origin of the problem. It is obvious from Section 4.3 that the set of these features and the classification must be as flexible as possible. A topic for our future work is to investigate if a finer theoretical classification (or set of features) similar to that for partial-differential equations from [27] is also useful (and not too inefficient) in our context. In [27], not as much attention is paid to solver-oriented features as to problem-centric ones. However, we also aim to recommend specific solver settings, which makes a careful study of IVPS characteristics necessary. At the moment, solvers are characterized mainly by their names, parameters (cf. Column 5 from Table 1) with their default values, and a textual description of their methods.

Our long-term (and somewhat ideal) goal is to provide a common environment for testing and recommending IVPS under the same conditions. This goal necessitates development of a semi-automatic procedure for adding a new solver to the solver database, which is a complex task for verified solvers since they lack common interfaces and are very different in their underlying concepts. A more manageable task would be to provide (template of) a solver database to be filled by an IVPS developer, which would leave common test conditions out the consideration. A useful feature in this case would be to convert the available benchmark sets (common ground, application or practice-oriented, see Section 4.2) into the syntax supported by a given solver or, at least, easy to use with it. Similar converters exist, for example, for hybrid systems (cf. HyST[11]). However, they seem to develop a new converter for each new solver. A more interesting approach would be to automatize this process along with the database generation itself, for example, through the use of XML specifications.

If we have a large (and extendable) set of solvers and want to adapt the problem classification in an easy way if necessary, we can consider descriptions in form of keywords. An approach retrieving information from relational and unstructured data might be useful as a filtering and structuring pre-step for the SRS routines described in Section 4.3. Following [51, 52], we assume that each request r for a solver v is defined by a keyword vector $r = (r_1, \dots r_J)^{\mathrm{T}}$ and a corresponding numeric vector $u = (u_1, \dots, u_J)^{\mathrm{T}}$, where $u_j = 1$ for each j-th keyword constituting the request r, $j = 1 \dots J$. Each tool v has a profile represented by a feature vector c_r built from keywords s_i and their weights ω_i^k, $i = 1, \dots, I$, $k = 1, \dots, K$. The weights influence the rating of the tool v w.r.t. each descriptor s_i and incorporate various factors (benchmarks, actuality, fitting to the problem dimensions). For example, they can reflect the degree of belief in s_i according to its provenance. The weights are normalized and aggregated over $k = 1 \dots K$ to an average weight $\omega_i(v)$ only if s_i is equal to a keyword from the request, $s_i = r_j$, resulting in J such weights. Those are used in the cosine similarity measure

[11] www.verivital.com/hyst/

$$d_v := \frac{\sum_{j=1}^{J} u_j \cdot \omega_j}{\sqrt{\sum_{j=1}^{J} \omega_j^2} \cdot \sqrt{\sum_{j=1}^{J} u_j^2}}, \tag{6}$$

that describes the similarity between the request vector r and the feature vector c_r characterizing the qualification of the tool v to solve the task. The final ranking is a descending list sorted according to d_v.

Consider a small illustration for this algorithm in our context. Suppose we are interested in the subset of seven items (solvers with their settings) from Table 2[12]. Assume further that we want to find a good solver for linear IVPs applicable in real time. That is, r consists of two keywords r_1 =“linear”, r_2 =“online” and $u = (1\ 1)^{\mathrm{T}}$. Let the database on the seven items contain the information shown in Table 4. Here, the weights $\omega_i^1 \in [0, 1]$ reflect how relevant or good the solver is w.r.t. the meaning of a given keyword, whereas the weights $\omega_i^2 \in [0, 1]$ show the degree of confidence in this assessment according to its provenance. For example, ω_i^2 can be set to one if tests were performed using a common environment (e.g., VERICOMP); to some other number less than one if the assessment is supplied by the developer of the tool; and still a smaller number if it is based on information extracted (automatically) from publications or similar. Accordingly, the weights ω_i^2 for RiOT with the order 11 are less than one since the tests for this setting were not carried out in VERICOMP. Suppose there is no information about how well Valencia 0.025 performs with linear systems. The last column of Table 4 shows the ratings obtained using formula (6). For example, the average weight for the keyword “linear” for RiOT 5 is $\frac{0.6+1}{2} = 0.8$ (we can omit normalization here), the weight vector consists of two components $(0.8\ 0.6)^{\mathrm{T}}$ and $d_{\text{RiOT 5}} = \dfrac{1.4}{\sqrt{0.8^2 + 0.6^2} \cdot \sqrt{2}} = 0.9899$ (truncated). Since a keyword is missing from the description of Valencia 0.025, no rating is generated. According to Table 4, the suitable solvers are Valencia with the stepsize 0.025 and VNODE with orders of Taylor expansion 15 and 20. This smaller set of solvers can be chosen for actual test runs, from the results of which recommender algorithms both from this section and Section 4.3 can profit. Better results can be achieved with more sophisticated normalizing and weighting strategies. Keywords and actual values of weights for them can be assigned on the basis of the previously computed quality criteria or such linguistic descriptions as ‘good’ or ‘bad’ extracted from papers. The results can be fed back to run tests and update the database.

5 CONCLUSIONS

In this paper, we presented the state of the art in the field of scientific recommender systems with a focus on visual analytics and problem-solving environments. We identified concepts, components and approaches necessary to recommend a scientific tool. Finally, we described in detail a possibility to recommend a solver for initial value problems depending on a user’s problem, partially supported by a testing environment VERICOMP accessible online. In particular, we discussed various quality criteria, metrics, problem classifications and problem/solver features based on an extensive literature study. The recommender algorithm was illustrated using a small example.

Our future work concerns implementing the discussed options in the common test environment VERICOMP. The possibility of manual filling of VERICOMP’s database with simulation results for a new solver by a registered user is under construction in its new version. Moreover, we plan to make the recommender available in the near future. Our middle-term goals

[12]Obviously, this pre-step makes more sense if there are many more solvers in the database

are to make the database construction more flexible by allowing it to be generated from XML-like descriptions and to provide templatized converters of benchmark sets to respective solvers' syntaxes.

REFERENCES

[1] *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2017.

[2] Ch. C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, 1st edition, 2016.

[3] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[4] M. Angelova, V. Devagiri, V. Boeva, P. Linde, and N. Lavesson. An expertise recommender system based on data from an institutional repository (DiVA). In *Proc. ELPUB*, Toronto, Canada, 2018.

[5] E. Auer. *Result Verification and Uncertainty Management in Engineering Applications*. Verlag Dr. Hut, 2014. Habilitation Monograph.

[6] E. Auer and A. Rauh. VERICOMP: a system to compare and assess verified IVP solvers. *Computing*, 94(2):163–172, Mar 2012.

[7] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, T. Schreck, D. Weiskopf, and D. A. Keim. Quality metrics for information visualization. *Computer Graphics Forum*, 37(3):625–662, 2018.

[8] B. W. Boehm. Some future software engineering opportunities and challenges. In *The Future of Software Engineering*, pages 1–32. Springer, Berlin, Heidelberg, 2011.

[9] S. Bönisch, M. Brickenstein, H. Chrapary, G.-M. Greuel, and W. Sperber. swMATH - A new information service for mathematical software. In *Intelligent Computer Mathematics - MKM, Calculemus, DML, and Systems and Projects 2013, Part of CICM 2013, Bath, UK, July 8-12, 2013. Proceedings*, pages 369–373, 2013.

[10] O. Bouissou and M. Martel. A Runge-Kutta method for computing guaranteed solutions of ODEs. In *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics, SCAN'06, Duisburg, Germany*, 2006.

[11] S. A. C. Bukhari, M. Martínez Romero, M. J. O'Connor, A. L. Egyedi, D. Willrett, J. Graybeal, M. A. Musen, K.-H. Cheung, and S. H. Kleinstein. CEDAR OnDemand: A browser extension to generate ontology-based scientific metadata. *BMC Bioinformatics*, 19(1):268:1–268:6, 2018.

[12] F. Bünger. Shrink wrapping for Taylor models revisited. *Numerical Algorithms*, 78(4):1001–1017, 2018.

[13] M. Capinski, J. Cyranka, Z. Galias, T. Kapela, M. Mrozek, and P. Zgliczynski. Computer assisted proofs in dynamics group. `capd.ii.uj.edu.pl`.

[14] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, June 2015.

[15] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 258–263, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[16] H. Chrapary and W. Dalitz. Software products, software versions, archiving of software, and swMATH. In *Mathematical Software - ICMS 2018 - 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings*, pages 123–127, 2018.

[17] M. Crubézy and M. A. Musen. Ontologies in support of problem solving. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 321–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[18] J. A. dit Sandretto and A. Chapoutot. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing electronic edition*, 22, July 2016.

[19] T. Dzetkulič. Rigorous integration of non-linear ordinary differential equations in Chebyshev basis. *Numerical Algorithms*, 69(1):183–205, May 2015.

[20] I. Eble. *Über Taylor-Modelle*. PhD thesis, Universität Karlsruhe, 2007.

[21] St. van den Elzen and J. J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 2013.

[22] A. Felfernig, G. Ninaus, H. Grabner, F. Reinfrank, L. Weninger, D. Pagano, and W. Maalej. An overview of recommender systems in requirements engineering. In *Managing Requirements Knowledge.*, pages 315–332. Springer, Berlin, Heidelberg, 2013.

[23] A. Geyer-Schulz, M.l Hahsler, and M. Jahn. Educational and scientific recommender systems: Designing the information channels of the virtual university. *International Journal of Engineering Education*, pages 153–163, 2001.

[24] E. Gómez-Martínez, M. Linaje, F. Sánchez-Figueroa, A. Iglesias-Pérez, J. C. Preciado, R. González-Cabero, and J. Merseguer. A semantic approach for designing assistive software recommender systems. *Journal of Systems and Software*, 104(C):166–178, June 2015.

[25] A. I. Guseva, V. S. Kireev, P. V. Bochkarev, I. A. Kuznetsov, and S. A. Philippov. Scientific and educational recommender systems. *AIP Conference Proceedings*, 1797(1):020002, 2017.

[26] M. Hernández-Rubio, I. Cantador, and A. Bellogín. A comparative analysis of recommender systems based on item aspect opinions extracted from user reviews. *User Modeling and User-Adapted Interaction*, Nov 2018.

[27] E. N. Houstis, A. C. Catlin, N. Dhanjani, J. R. Rice, N. Ramakrishnan, and V. S. Verykios. MyPYTHIA: A recommendation portal for scientific software and services. *Concurrency and Computation: Practice and Experience*, 14(13-15):1481–1505, 2002.

[28] E. N. Houstis, A. C. Catlin, J. R. Rice, V. S. Verykios, N. Ramakrishnan, and C. E. Houstis. PYTHIA-II: A knowledge/database system for managing performance data and recommending scientific software. *ACM Trans. Math. Softw.*, 26(2):227–253, June 2000.

[29] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick. Comparing Numerical Methods for Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[30] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[31] F. Immler. *A Verified ODE Solver and Smale's 14th Problem*. PhD thesis, TU Müchen, 2018.

[32] D. Jannach, M. Zanker, M. Ge, and M. Gröning. Recommender systems in computer science and information systems – a landscape of research. In Christian Huemer and Pasquale Lops, editors, *E-Commerce and Web Technologies*, pages 76–87, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[33] M. Kashiwagi. Verified numerical computation and `kv` library. `http://verifiedby.me/kv/index-e.html`.

[34] P. Kaur and M. Owonibi. A review on visualization recommendation strategies. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, pages 266–273, 2017.

[35] A. Key, B. Howe, D. Perry, and C. Aragon. VizDeck: Self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 681–684, New York, NY, USA, 2012. ACM.

[36] A. Klahold. *Empfehlungssysteme: Grundlagen, Konzepte und Lösungen*. Vieweg Teubner, 2009. In German.

[37] D. E. Knuth. Literate Programming. *The Computer Jour.*, 27(2):97–111, 1984.

[38] B. Latard, J. Weber, G. Forestier, and M. Hassenforder. Towards a semantic search engine for scientific articles. In *Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings*, pages 608–611, 2017.

[39] F. Lorenzi and F. Ricci. Case-based recommender systems: A unifying view. In Bamshad Mobasher and Sarabjot Singh Anand, editors, *Intelligent Techniques for Web Personalization*, pages 89–113. Springer Berlin Heidelberg, 2005.

[40] J. Mackinlay, P. Hanrahan, and Ch. Stolte. Show Me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, November 2007.

[41] A. Mahboubi, G. Melquiond, and Th. Sibut-Pinote. Formally verified approximations of definite integrals. *J. Autom. Reasoning*, 62(2):281–300, 2019.

[42] K. Makino and M. Berz. Suppression of the wrapping effect by Taylor model-based validated integrators. MSUHEP 40910, Department of Physics, Michigan State University, East Lansing, MI 48824, 2004.

[43] N. Manouselis and C. Costopoulou. *Personalization Techniques and Recommender Systems*, chapter Experimental Analysis of Multiattribute Utility Collaborative Filtering on a Syntetic Data Set, pages 111–133. World Scientific Publishing Company, 2008.

[44] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[45] B. Mutlu, E. Veas, and Ch. Trattner. VizRec: Recommending personalized visualizations. *ACM Trans. Interact. Intell. Syst.*, 6(4):31:1–31:39, November 2016.

[46] K. Nazemi, R. Retz, J. Bernard, J. Kohlhammer, and D. W. Fellner. Adaptive semantic visualization for bibliographic entries. In *Advances in Visual Computing - 9th International Symposium, ISVC 2013, Rethymnon, Crete, Greece, July 29-31, 2013. Proceedings, Part II*, pages 13–24, 2013.

[47] N.S. Nedialkov. *Implementing a Rigorous ODE Solver Through Literate Programming*, volume 3 of *Mathematical Engineering*, pages 3–19. Springer, Heidelberg, 2011.

[48] A. W. Neumann. *Recommender Systems for Information Providers – Designing Customer Centric Paths to Information*. Springer, 2009.

[49] Ch. Obeid, I. Lahoud, H. El Khoury, and P.-A. Champin. Ontology-based recommender system in higher education. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1031–1034, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[50] D. H. Park, I. Y. Choi, H. K. Kim, and J. K. Kim. A review and classification of recommender systems research. In *International Conference on Social Science and Humanity IPEDR vol. 5*, pages 290–294, Singapore, 2011. IACSIT Press.

[51] J. Protasiewicz. Support system for selection of reviewers. In *IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 3062–3065, San Diego, CA, USA, 2014.

[52] J. Protasiewicz, W. Pedrycz, M. Kozlowski, S. Dadas, T. Stanislawek, A. Kopacz, and M. Galźźewska. A recommender system of reviewers and experts in reviewing problems. *Know.-Based Syst.*, 106(C):164–178, August 2016.

[53] A. S. Raamkumar, S. Foo, and N. Pang. Can I have more of these please? assisting researchers in finding similar research papers from a seed basket of papers. *The Electronic Library*, 36(3):568–587, 2018.

[54] A. Rauh and E. Auer. *Modeling, Design, and Simulation of Systems with Uncertainties*, volume 3 of *Mathematical Engineering*. Springer, Heidelberg, 2011.

[55] A. Rauh, E. Auer, J. Minisini, and E. P. Hofer. Extensions of VALENCIA-IVP for Reduction of Overestimation, for Simulation of Differential Algebraic Systems, and for Dynamical Optimization. In *Proceedings in Applied Mathematics and Mechanics*, pages 1023001–1023002, 2007.

[56] M. P. Robillard, W. Maalej, R. J. Walker, and Th. Zimmermann. *Recommendation Systems in Software Engineering*. Springer Publishing Company, 2014.

[57] J. Scholtz, C. Plaisant, M. A. Whiting, and G. G. Grinstein. Evaluation of visual analytics environments: The road to the visual analytics science and technology challenge evaluation methodology. *Information Visualization*, 13(4):326–335, 2014.

[58] S. Siebert, S. Dinesh, and S. Feyer. Extending a research-paper recommendation system with bibliometric measures. In *Proceedings of the Fifth Workshop on Bibliometric-enhanced Information Retrieval (BIR) co-located with the 39th European Conference on Information Retrieval (ECIR 2017), Aberdeen, UK, April 9th, 2017.*, pages 112–121, 2017.

[59] L. Steinert. *Beyond Similarity and Accuracy - A New Take on Automating Scientific Paper Recommendations*. PhD thesis, University of Duisburg-Essen, Germany, 2017.

[60] Ch. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional databases. *Commun. ACM*, 51(11):75–84, November 2008.

[61] M. Thurau, Chr. Buck, and W. Luther. IPFViewer: Incremental, approximate analysis of steel samples. In *Proceedings of SIGRAD 2014, Visual Computing, June 12-13, 2014, Göteborg, Sweden*, pages 1–8, 2014.

[62] S. W. Tu, H. Eriksson, J. H. Gennari, Y. Shahar, and M. A. Musen. Ontology-based configuration of problem-solving methods and generation of knowledge-acquisition tools: application of PROTEGE-II to protocol-based decision support. *Artificial Intelligence in Medicine*, 7(3):257–289, 1995.

[63] W. Tucker. The Lorenz attractor exists. *C. R. Acad. Sci. Paris Sér. I Math.*, 328(12):1197–1202, 1999.

[64] W. Tucker. A rigorous ODE solver and Smale's 14th problem. *Found. Comput. Math.*, 2(1):53–117, 2002.

[65] M. Voigt, M. Franke, and K. Meissner. Using expert and empirical knowledge for context-aware recommendation of visualization components. *IARIA Int. Jour. on Advances in Life Sciences*, 5(1):27–41, 2013.

[66] B. Weyers, E. Auer, and Luther W. The role of verification and validation techniques within visual analytics. *JUCS: Special Issues on Collaborative Technologies and Data Science in Smart City Applications*, 2019. submitted.

[67] D. C. Wilson, D. B. Leake, and R. Bramley. Case-based recommender components for scientific problem-solving environment. In *Proceedings of the Sixteenth IMACS World Congress*, 2000.

[68] K. Wongsuphasawat, D. Moritz, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Trans. Vis. Comput. Graph.*, 22(1):649–658, 2016.

[69] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017.*, pages 2648–2659, 2017.

[70] S. Zhao, D. Zhang, Z. Duan, J. Chen, Y.-P. Zhang, and J. Tang. A novel classification method for paper-reviewer recommendation. *Scientometrics*, 115(3):1293–1313, June 2018.

IVPS	Lang.	VC	Main method	Main parameters
Valencia [55]	C++	RV	Picard iteration, exponential extension	stepsize
VNODE [47]	C++	RV LP	Taylor series (TS), Hermite-Obreschkoff method (HO)	method/ order/ tolerances/ min.stepsize/ stepsize control
CAPD [13]	C++	RV	TS, explicit-implicit HO	method/order/tolerances
COSI-VI [42]	FORTRAN	RV	Taylor models	TM order/ stepsize/ tolerances/ preconditioning/ shrink wrapping
RiOT [20]	C++	RV	Taylor models (TM)	TM order/ bounding method/ stepsize control/ tolerances/ sparsity
verifyode [12]	INTLAB	RV	Taylor models	TM order/ bounder/stepsize control/ tolerances/shrink wrap./ sparsity
Flow* [15]	C++	RV	Taylor models	TM order/stepsize control/ tolerances/no.steps with symbolic remainders/ remainder estimation bound
kv [33]	C++	RV	Power series and affine arithmetics	method/ma precision
[19]	C++	RV	TM, Chebyshev function enclosures	method/ order/ sparsity/ tolerances
DYNIbex [18]	C++	RV	affine arithmetic, Runge-Kutta	RK variant/ order/ stepsize/ tolerances
GRK [10]	OCaml	RV	Runge-Kutta, multiprecision arithmetics	stepsize control/ tolerances
CORA [3]	MATLAB	RV	Reachability analysis with zonotopes/ polytopes	(polynomial) zonotope order/ tolerances
Isabelle [31]	PolyML	FV RV	Affine arithmetic/ zonotopes, Runge-Kutta	max. zonotope order/ stepsize control/ tolerances/ ma precision
[41]	Coq	FV RV	antiderivatives of rigorous polynomial approximations, adaptive domain splitting	ma precision/ target error bound

Table 1: Verified IVPS. ‘VC’ means verification concept, with possibilities ‘RV’ (result verification), ‘LP’ (literal programming), ‘FV’ (formal verification). The abbreviation ‘ma’ means ‘machine arithmetic’, usually floating point.

Solver	u_1					u_2				
	t_{c}	t_{us}	e_{us}	t_{bd}	e_{bd}	t_{c}	t_{u}	e_{u}	t_{bd}	e_{bd}
RiOT 5	3.270s	3.197s	0.448	10	0.130	3.597s	3.466s	0.811	10	0.20
RiOT 10	13.030s	12.763s	0.443	10	0.057	0.860s	0.842s	0.811	10	0.20
RiOT 15	40.883s	40.607s	0.443	10	0.055	0.918s	0.886s	0.811	10	0.20
V 0.025	0.045s	0.042s	2.987	1.300	5.85	0.260s	0.257s	0.850	10	309.55
V 0.0025	0.287s	0.282s	2.905	1.17	3.69	1.528s	1.521s	0.815	10	249.32
V 0.00025	2.794s	2.780s	2.897	1.19	3.77	1m30.844s	1m30.726s	0.812	10	243.87
VNODE 15	0.014s	0.009s	0.887	6.36	151.77	0.047s	0.041s	0.811	10	0.203
VNODE 20	0.014s	0.007s	0.987	3.81	218.18	0.047s	0.042s	0.811	10	0.203
VNODE 25	0.015s	0.009s	1.138	2.59	270.42	0.046s	0.039s	0.811	10	0.203

Table 2: Test run data on the problems u_1, u_2 from the database of the old version of VERICOMP. V stands for Valencia. **C4** is defined by t_{c}, **C5** by t_{us} and e_{us}, and **C6** by t_{bd} and e_{bd}. Besides, $t_{\text{out}} = 1\text{s}$ and $t_{\text{max}} = 10\text{s}$.

v_i	$C_4(v_i, u_i)$ (u_1/u_2)	$C_5(v_i, u_i)$ (u_1/u_2)	$C_6(v_i, u_i)$ (u_1/u_2)	$\frac{\chi(v_i,u_1)+\chi(v_i,u_2)}{2}$
RiOT 5	0.007/0.007	0.007/0.007	0.936/0.500	0.149
RiOT 10	0.007/0.008	0.007/0.008	0.999/0.500	0.155
RiOT 15	0.007/0.007	0.007/0.008	0.999/0.500	0.155
V 0.025	0.058/0.009	0.014/0.010	0.007/0.007	0.020
V 0.0025	0.009/0.007	0.008/0.007	0.007/0.007	0.007
V 0.00025	0.007/0.007	0.007/0.007	0.007/0.007	0.006
VNODE 15	0.89/0.05	0.99/0.11	0.01/0.48	0.462
VNODE 20	0.89/0.05	0.99/0.11	0.01/0.48	0.461
VNODE 25	0.84/0.05	0.99/0.13	0.01/0.48	0.453

Table 3: Recommendation process with three criteria and nine solvers, based on the problems u_1 and u_2, using the normalizing function $n_{40,10}$ and rating averaging.

Solver	Keyword	ω_1^1	ω_1^2	Keyword	ω_2^1	ω_2^2	Keyword	ω_3^1	ω_3^2	Rating
RiOT 5	linear	0.6	1	nonlinear	0.8	1	online	0.2	1	98.99
RiOT 10	linear	0.5	1	nonlinear	0.9	1	online	0.1	1	98.83
RiOT 11	linear	0.5	0.6	nonlinear	0.95	0.6	online	0.1	0.6	97.61
V 0.025				nonlinear	0.2	1	online	0.6	1	–
V 0.0025	linear	0.6	1	nonlinear	0.2	1	online	0.56	1	99.99
VNODE 15	linear	0.9	1	nonlinear	0.7	1	online	0.7	1	99.84
VNODE 20	linear	0.95	1	nonlinear	0.8	1	online	0.6	1	99.51

Table 4: Solver descriptions in terms of feature vectors c_r. Ratings are multiplied by the factor 100 for better presentation.