International Journal of Combinatorial Optimization Problems and Informatics, 16(3), May-Aug 2025, 146-169. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2025.v16i3.587

www.editada.org

Multi-Agent Architecture for Parameter Configuration in Computational Chemistry

José Ángel Villarreal-Hernández, María Lucila Morales-Rodríguez, Nelson Rangel-Valdez,
Laura Cruz-Reyes, Claudia Gómez-Santillán
TecNM/Instituto Tecnológico de Ciudad Madero, Tamaulipas, México.
{D10071371, lucila.mr, nelson.rv, laura.cr, claudia.gs}@cdmadero.tecnm.mx

Abstract. Chemists often rely on computational prediction techniques to optimise laboratory experiments. However, selecting and configuring these techniques can be a daunting task for those without a background in computer science. A potential solution arises when chemists collaborate with computer scientists to define and implement the prediction task. Based on this collaboration, we propose a novel architecture that reflects key aspects of this interaction: the expertise of computer scientists, their commitment to addressing the prediction task, and the discussions that facilitate consensus. Our proposed architecture enables chemists to generate solution proposals across the various stages of the prediction process, leveraging consensus-based decisions made by intelligent agents and incorporating a lazy learning approach.

Keywords: Chemical prediction, Parameter setting, Software architecture, Multi-agent negotiation.

Article Info
Received November 29, 2024
Accepted February 18, 2025

1 Introduction

Computational chemistry is a scientific discipline dedicated to extracting structural and chemical information from molecular systems using mathematical calculations derived from fundamental physical laws (E. San Fabián, 2020). Despite the development of numerous software tools in this field, these tools generally fall into three primary application contexts: input representation, data presentation, and data processing. This paper highlights examples of computational chemistry contributions organized by these contexts.

In the context of input representation, significant contributions include chemical notation systems such as the Simplified Molecular Input Line Entry System (SMILES) (Weininger, 1988), which leverages molecular graph theory and is widely supported by software applications. Another example is the Crystallographic Information File (CIF) (Hall et al., 1991) , a self-descriptive format designed for both human and machine readability to manage crystallographic data. Input representation primarily addresses the capture and portability of data for subsequent processing and presentation.

In the data presentation context, contributions are centered on visualization techniques. For instance, Jmol (Hanson, 2016; Scalfani et al., 2016) is a molecular structure viewer that supports biochemical applications, employs 3D graphics, is compatible with SMILES, and operates on the HTML5 web standard. Visualization tools such as Jmol facilitate chemical research by enabling researchers to effectively capture and interpret structural information.

The data processing context encompasses software that applies computational methods to simulate molecular interactions and predict chemical properties. Examples include Gaussian (M. J. Frisch et al., 2016; Young, 2002) and GAMESS (Barca et al., 2020; Gordon & Schmidt, 2005), which implement various ab initio, molecular mechanics, and quantum mechanics methods. These tools are designed to predict reaction outcomes and material properties through computational modeling and simulation. Beyond computational calculus, modern approaches have integrated data mining and machine learning to enhance chemical

predictions. Data mining has been applied to predict trends and solve complex problems in chemical, material, and engineering processes (Hao Li et al., 2019). Machine learning techniques have been utilized for protein structure prediction, material design, and property prediction, offering classification and prediction capabilities (Goh et al., 2017).

Despite these advancements, the task of configuring computational prediction techniques often falls to chemists, who may lack formal training in computer science. This limitation leads to the treatment of prediction tools as opaque "black boxes" and contributes to suboptimal parameter configurations (Keith et al., 2021). The choice and configuration of algorithms, parameters, and datasets are critical to achieving reliable results but can present significant challenges without computational expertise. For example, consider a scenario where a chemist aims to predict whether the interaction between two substances will result in gel formation, solubility, insolubility, or precipitation. The chemist must define the dataset and problem characteristics and consult computational experts, each of whom will draw from their experience to propose appropriate parameter configurations. Differences in expertise and methodology often lead to varied solutions, underscoring the need for a systematic approach to parameter configuration.

This paper proposes an architecture designed to automate the configuration of predictive applications in computational chemistry. By simulating the reasoning processes of computational experts, the architecture optimizes the performance of machine learning and data mining techniques. The configuration process begins with a description of the prediction problem and relevant dataset characteristics, generating solution proposals tailored to the specific context. These solutions include the selection of computational techniques, parameter adjustments, and dataset modifications.

The proposed architecture employs a multi-agent system (MAS) to evaluate and propose parameter configurations. Each agent represents a virtual expert with distinct reasoning approaches, using attributes such as exploration and experience to adapt and improve their recommendations over time. The agents negotiate to reach consensus, ensuring that the solutions reflect collective expertise and are optimized for the prediction task.

The architecture features two operational phases: preparation and configuration. The preparation phase leverages an experiment generator to produce knowledge for the agents, allowing seamless integration of new data without requiring updates to a generalization model. This contrasts with architectures such as that proposed by (Cerecedo-Cordoba et al., 2017), which rely on intensive training phases to construct generalization models. While systems like the Automated Machine Learning (AutoML) (Hutter et al., 2019) focus on automating the entire machine learning pipeline—including model selection, hyperparameter optimization, and evaluation—our architecture emphasizes simulating the reasoning processes of computing experts in addressing prediction problems. In particular, our lazy learning approach allows knowledge to be incorporated from diverse sources, such as data mining on literature or procedural generation of artificial instances. In the configuration phase, the agents apply similarity-based reasoning, comparing examples in the training dataset with the situation presented by the end user. Unlike AutoML, which relies on predefined algorithms and optimization techniques, our architecture uses agents that represent different computer experts. These agents propose solutions using distinct approaches and negotiate to reach a consensus. This negotiation ensures that the solutions are acceptable to all agents and reflect the collective expertise embedded in the system.

This is followed by a review of algorithm configuration for computational chemistry, which explores key aspects such as prediction techniques (2.1), parameter adjustment challenges (2.2), and approaches to parameter setting (2.3). Next, an architecture for parameter configuration is proposed, including scenario analysis (3.1), multi-agent negotiations for collaborative decision-making (3.2), and the detailed architecture and its implementation (3.3). A proof of concept is then presented to validate the proposed architecture experimentally. The discussion section reflects on the results, their implications, and limitations, while the document concludes by summarizing the findings and suggesting directions for future work.

2 Review of algorithm configuration for computational chemistry

The deployment of prediction tasks in computational chemistry involves decision-making across various technical and theoretical dimensions. Examining the algorithms applied within this field provides insight into parameter adjustment strategies that can automate decision-making processes. Reviewing contributions to computational chemistry prediction facilitates the identification of key action points for the configuration architecture proposed in Section 3.

2.1 Prediction in computational chemistry

The prediction of chemical structures and properties is a well-explored topic in the literature, approached through diverse methodologies. For example, Finocchi (Finocchi, 2011) highlights theoretical tools such as Density Functional Theory (DFT), which serves as a computational alternative to the Schrödinger equation in materials science. Additionally, data mining and machine learning techniques have been extensively utilized to develop predictive systems in this domain.

Machine learning, a branch of artificial intelligence, is dedicated to identifying patterns and generalizations within datasets. It is defined as the study of algorithms that improve their performance with experience and data (Mitchell, 1997). Its purpose is to build computer programs that automatically shape and enhance your experience (Stuart J. Russell et al., 2010). Conversely, data mining is a methodology for uncovering intrinsic patterns and relationships within data, often for predictive purposes. By leveraging statistical analysis, data mining predicts variables that are otherwise challenging to obtain experimentally (Hao Li et al., 2019). This approach is closely associated with computational statistics, particularly in unsupervised learning contexts, and is often applied in exploratory data analysis (Fehri et al., 2019).

Data mining techniques have demonstrated significant success in predicting trends and solving complex problems in chemical, material, and engineering processes (Hao Li et al., 2019). For example, Günay (M. Erdem Günay et al., 2018) employed decision trees to identify optimal conditions for CO_2 electroreduction, using the technique to determine variable significance and identify three key traits of interest.

Machine learning has been applied to a wide range of problems in chemistry. For instance, Gupta (Gupta et al., 2016) explored gelation prediction using various machine learning techniques, including Support Vector Machines (supervised learning), Random Forest (ensemble learning), Artificial Neural Networks (ANN, supervised learning), and K Nearest Neighbors (KNN, pattern recognition). Similarly, Loredo's work (Loredo-Pong et al., 2022) focused on predicting the gelation of benzoate organogelators using the KNN algorithm. Their study involved generating training datasets with alternating physicochemical attributes, including polarity, heteroatoms, and solvent saturation. The classifier's output—a predicted aggregation state (gel, soluble, insoluble, or precipitate)—was validated against laboratory results.

Deep learning has experienced a resurgence in computational chemistry, as noted by Goh (Goh et al., 2017). The author highlights the efficiency and applicability of deep neural networks in areas such as quantitative structure-activity relationships, protein structure prediction, material design, and property prediction. This resurgence is attributed to advancements in GPU-based computation and the expansion of chemical data for training purposes.

A notable example of integrating data mining and machine learning is the work by Cerecedo (Cerecedo-Cordoba et al., 2017), who proposed an architecture for estimating the melting temperature of ionic liquids. Their approach includes an extensive training phase involving clustering techniques, genetic algorithms to generate artificial neural networks (ANNs), and cross-validation for result weighting and parameter selection.

These works can be conceptually described through a three-layer framework (see Figure 1). The base layer addresses the real-world prediction problem, while the middle layer focuses on problem treatment, including algorithm selection and dataset structure. The upper layer involves parameter selection, specifying the configuration values that optimize algorithm performance.

For example, Loredo addressed the prediction of gelation (base layer) using the KNN algorithm (problem treatment layer) and obtained parameter values through experimental design (parameter selection layer). Figure 1 illustrates this conceptual framework. Similarly, Cerecedo tackled the prediction of melting temperatures (base layer) using neuroevolution techniques (problem treatment layer) and an exhaustive grid search for parameter optimization (parameter selection layer).

	Case A	Case B	Case C
Parameter choice layer	Design of experiments	Exhaustive grid search	Multi-agent system adjusts algorithm parameters
Problem treatment layer	K nearest neighbors	Neuroevolution	Multi-agent system chooses algorithm and data body
Real problem layer	Prediction of benzoate organogelator gelation	Prediction of melting temperature of imidazole ionic liquids	To be defined by the user

Fig. 1. Layer diagram, application case abstraction. The configuration architecture acts on all three layers of the prediction task

Like other computational techniques, data mining and machine learning algorithms require users to define specific configurations to ensure optimal performance. For instance, in the KNN algorithm, the parameter k determines the number of nearest neighbors considered during classification. The performance of these algorithms is intrinsically linked to their parametric configuration, and various optimization techniques have been proposed in the literature to address this challenge.

2.2 Parameter adjustment

The parameter adjustment problem pertains to the selection of appropriate values for algorithm parameters to control their behavior effectively (Birattari, 2005). It constitutes a computational optimization problem, as the efficiency and performance of algorithms are significantly influenced by their parametric configurations. The specific combination of parameter values for an algorithm is referred to as its parametric configuration.

Identifying optimal parameter values is critical, particularly for specific problems and contexts. However, as noted by Molina (Molina et al., 2012), djusting the parameters of data mining techniques can be computationally expensive and may introduce biases due to assumptions made during the process. Automatic parameter adjustment has been widely studied in the literature, employing various approaches and techniques. For instance, Cayci's work (Cayci et al., 2011), utilized Bayesian networks to identify the effects of parameter configurations in data extraction algorithms. Similarly, Srivastava (Srivastava, 2005) employed learning techniques to optimize parameter configurations in search-based algorithms, focusing on domain-specific adjustments and incorporating user-defined criteria.

Michalewicz (Michalewicz & Fogel, 2004) proposes a classification of parameter setting strategies based on the level of adaptation, as illustrated in Figure 2. This classification distinguishes between two primary moments of adjustment:

- Parameter Tuning: This refers to adjustments made before the algorithm's execution, providing an initial configuration. Techniques for parameter tuning include trial and error, experimental design, and metaheuristic algorithms, which iteratively optimize parameter values.
- Parameter Control: This involves dynamic adjustments made during the algorithm's execution. Parameter control can be further categorized as:
  - Deterministic Control: The parametric configuration follows predefined rules, without feedback from the algorithm's performance.
  - Adaptive Control: Adjustments are informed by feedback, allowing parameters to change in response to the algorithm's state.
  - Self-Adaptive Control: Parameters are encoded within the algorithm itself and evolve through mechanisms such as mutation or recombination. For example, in genetic algorithms, parameter values are represented within chromosomes and optimized alongside solutions.

Although parameter tuning and control are often treated as separate processes, Eiben (Eiben et al., 1999). recommend combining these strategies—using tuning techniques to determine initial configurations and parameter control to adapt settings during execution.

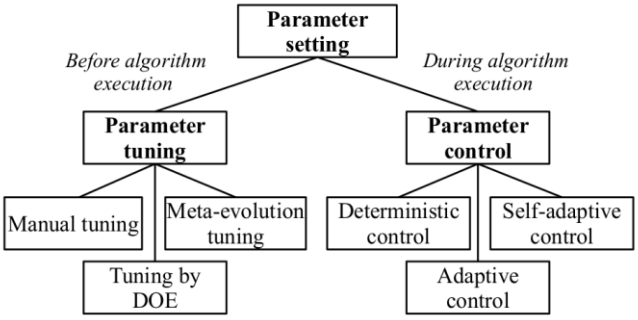

Fig. 2. Classification of parameter setting (Michalewicz & Fogel, 2004)

The study of parameter adjustment examines the influence of algorithmic parameters on results, aiming to optimize configurations while avoiding overfitting. Overfitting occurs when parameters are excessively tailored to a specific dataset, potentially compromising generalizability. Eiben (Eiben et al., 1999) provide four key guidelines for selecting appropriate control techniques:

- Parameters exhibit varying degrees of influence on algorithm performance; therefore, understanding the parameter-impact relationship is essential.
- Problem-specific characteristics influence parameter effectiveness, making it necessary to identify those that render parameter values suboptimal and may require adjustment during execution.
- The role of each parameter in eliciting desired algorithmic behavior should be carefully studied, along with the algorithm's operational states.
- Relevant information—such as problem-specific data, algorithmic learning insights, execution statistics, and runtime status—should inform the generation of new parametric configurations.

By systematically addressing these considerations, it is possible to refine parameter adjustment strategies, enhancing the performance and reliability of algorithms across diverse applications.

2.3 Techniques for parameter setting

Heuristic and evolutionary computation techniques have been widely employed to address the parameter adjustment problem. For instance, Martínez (Martínez González, 2001) demonstrated the use of heuristic methods, specifically the simplex method, to optimize parameter settings for various models. The study included simulations of a desorption column using Ecosim software, highlighting the effectiveness of the Nelder-Mead simplex method, particularly in scenarios involving numerous parameters.

Regarding metaheuristic techniques—general strategies for designing algorithms—Rivera (Rivera Zárate, 2009) investigated parameter adjustment under dynamic conditions for the semantic query routing problem (SQRP). Using an ant colony optimization algorithm, Rivera integrated Hoeffding competencies to configure the algorithm, achieving superior performance for SQRP in comparison to other approaches documented in the literature.

Evolutionary computation has been applied to parameter setting, as demonstrated by Fernández (Fernandez et al., 2019). Their work tackled parameter elicitation for multi-criteria decision problems using an evolutionary algorithm (EA). Starting with a preference model and a set of examples provided by decision-makers, the study employed Preference Disaggregation Analysis (PDA), a specialized regression method, to estimate parameter values. Fernández optimized PDA results using a multi-objective genetic algorithm (GA), demonstrating how indirect elicitation enables parameter inference from example datasets.

Cerecedo (Cerecedo-Cordoba et al., 2017) presented an architecture for estimating the melting temperature of ionic liquids. The methodology involved an exhaustive grid search to configure Support Vector Regression (SVR) and Regression Trees (RT), clustering data into groups. Subsequently, different artificial neural networks (ANNs) were trained and optimized using genetic algorithms. The approach aimed to enable ANNs to effectively learn the selection of SVR and RT models, contributing to improved predictive accuracy.

Automated Machine Learning (AutoML) research, as summarized by Hutter (Hutter et al., 2019), seeks to automate the end-to-end application of machine learning to real-word problems. Core tasks include feature engineering, model selection, hyperparameter tuning, and evaluation. Techniques such as reinforcement learning, Bayesian optimization and evolutionary algorithms are frequently used. AutoML has shown notable success in domains such as healthcare, finance, and natural language processing, reducing manual effort while delivering robust, data-driven insights.

Referring to the abstraction in Figure 1, the works described align with the parameter selection layer. These contributions share the goal of optimizing parameter values to enhance algorithm performance. However, several challenges emerge:

- Parameter Interdependence: Some methods adjust all parameters simultaneously, treating them as a unified entity without distinguishing their individual roles in algorithmic operations. This approach requires balancing the roles of each parameter to achieve satisfactory algorithm performance. While multiple configurations may be generated and the most suitable one selected, this process can be time-consuming and reliant on user expertise.
- Single Parameter Focus: Methods targeting individual parameter adjustments often require functions that consider interactions with other parameters. This shifts the challenge of balance to the parameters involved in these interdependencies.
- Lack of Experience Mechanisms: Selecting optimal configurations often relies on expert experience, which includes understanding algorithmic operations and research objectives while avoiding biases. However, current state-of-the-art methods lack mechanisms to accumulate and leverage experience over time. Instead, parameter adjustment is frequently treated as an isolated exercise tailored to specific research scenarios.

Cerecedo (Cerecedo-Cordoba et al., 2017), illustrate a potential solution by generating multiple configurations, prioritizing the most promising, and discarding less effective ones. While experience does not eliminate the need for running algorithms with varied settings, it can accelerate the identification of relevant configurations and reduce wasted computational resources. Developing mechanisms to capture and apply accumulated knowledge could significantly enhance the efficiency of parameter adjustment processes.

3 An architecture for parameter configuration

3.1 Scenario analysis

In computational chemistry, opportunities arise in improving the performance and behavior of data mining and machine learning techniques used for prediction. Many of these techniques require adapting parameters to specific application scenarios. However, users without expertise in computer science, such as chemists, often rely on advice from computer scientists to configure these parameters.

Computer science experts typically recommend techniques and parameter settings based on their experience. Yet, different experts may propose different solutions, reflecting varying commitments or preferences. These differences may lead to varying levels of practical effectiveness. This scenario suggests the potential for modeling expert interactions using an agent paradigm, where experts (agents) negotiate and collaborate to automate consensus decision-making for optimal parameter configuration.

Figure 3 illustrates an overview of the architecture for parameter configuration, which comprises two primary phases: the preparation phase and the configuration phase. The preparation phase focuses on building a experience corpus about predictive techniques and their behavior under various conditions. To generate this knowledge, an experiment generator performs exploratory runs, recording the results and relevant characteristics in a database. In the configuration phase configurators utilize the experience corpus to analyze the user's query and propose configurations for the prediction task. The configurators engage in a negotiation process to refine these proposals and reach a consensus. The outcome is a set of configurations tailored to the task, aligned with the negotiation and interaction strategies detailed in section 3.2.

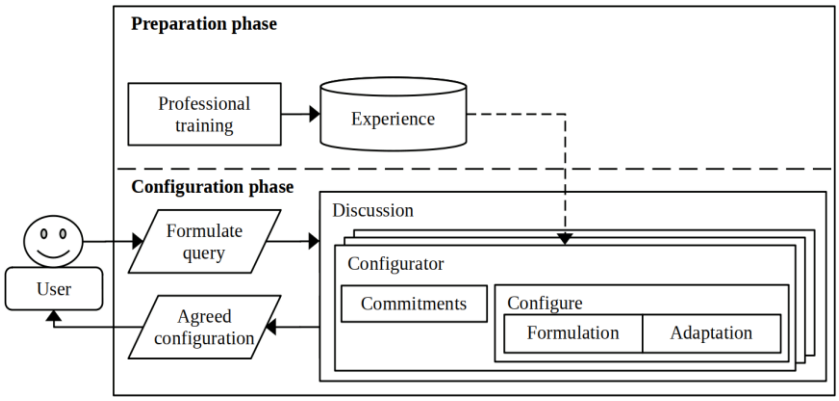

Fig. 3. General view of the architecture for parameter configuration

The architecture integrates seven key elements:

- User's query: Defines the problem and provides the associated data corpus.
- Configurators: Represent expert agents tasked with proposing configurations.
- Configurators' experience: An objective and standardized experience corpus derived from observed algorithm behavior.
- Configuration commitments: Tendencies influencing how configurators approach prediction tasks.
- Formulation of configurations: Internal process where configurators identify solutions based on prior experience.
- Discussion of configurations: Negotiation among configurators to evaluate and refine proposals.
- Adaptation of configurations: Mechanism enabling configurators to revise configurations during discussions.

The user's query serves as the input for the architecture, describing the prediction problem and the associated data corpus. The user must capture characteristics of the prediction problem using a form that covers aspects relevant to the configurators. For example, configurators may approach problems differently depending on whether variable selection has already been performed on the dataset.

Configurators' experience consists of a systematically built experience corpus derived from observations of algorithm performance under diverse conditions. To minimize bias, this experience is designed to be broad, objective, and uniformly applied across all configurators.

The formulation of configurations involves configurators analyzing their experience to locate past examples most similar to the user's query. This process enables them to generate initial solutions tailored to the problem.

Once configurations are formulated, the discussion phase begins. Configurators propose solutions and collaboratively evaluate their suitability. Through this negotiation, they refine the proposals, identifying configurations that best address the prediction task.

The adaptation of configurations occurs during discussions when configurators revise their initial proposals. This iterative process ensures that configurators can incorporate feedback and converge on improved solutions. The extent of these adaptations depends on each configurator’s willingness to adjust their initial preferences.

Finally, configuration commitments represent the tendencies or preferences that influence how configurators approach prediction tasks. These commitments can focus on algorithm selection, parameter tuning, or dataset modifications. Each commitment shapes how configurators prioritize features and assess their relevance to the user's query, directly impacting the configuration process.

Figure 1 (Case C) illustrates the architecture's abstraction. The user defines the real problem layer, while the architecture addresses the parameter selection and problem treatment layers. Based on the user's data, the architecture evaluates algorithm options, parametric configurations, and dataset modifications to optimize the prediction task.

The negotiation and adaptation of configurations among configurators represent a challenge in itself. The final configurations must satisfy all parties while combining the strengths of individual proposals. Viewing configurators as a multi-agent system (MAS) provides a solution framework. MAS enables negotiation and collaboration, employing computational techniques for parameter tuning and leveraging machine learning to improve configurators' experience and results over time.

The described architecture demonstrates potential for automating parameter configuration, reducing reliance on human expertise, and enhancing predictive performance. Further development of MAS-based negotiation mechanisms and experience-driven learning will enable more robust and efficient solutions tailored to diverse scientific challenges.

3.2 Multi-agent negotiations

Negotiation plays a fundamental role in various domains of human interaction. While often associated with formal agreements such as commercial transactions, labor contracts, or political treaties, negotiation occurs in everyday scenarios, often without explicit acknowledgment of its presence. For instance, a family deciding whether to spend a Sunday afternoon at a park or at the cinema engages in a negotiation process. Factors influencing this decision may include individual preferences for activities or specific information, such as available movie showings. Similarly, subsequent decisions—such as where to eat—constitute additional negotiation scenarios, balancing options like dining at a restaurant or ordering specific types of food. These examples illustrate that negotiation serves as a mechanism to reconcile differing interests and reach solutions that are both realistic and mutually satisfactory. Formally, negotiation is a structured process aimed at achieving agreements, whether for resolving conflicts, establishing partnerships, or optimizing collaborative outcomes (Baarslag, 2014).

In the context of Multi-Agent Systems (MAS), negotiators are represented by deliberative software agents. Endowing these agents with negotiation capabilities necessitates engagement with the field of automated negotiation, which focuses on the formalization of negotiation mechanics and strategies into computable models. The goal is to develop algorithms and protocols that facilitate effective agreement formation.

Iglesias (Iglesias Fernández, 1998) identifies three fundamental aspects integral to negotiation in MAS:

- Communication Mechanisms: These define the language and semantics for exchanging negotiation-related information, such as proposals, refinements, and confirmations, as well as the objects of negotiation (e.g., plans, task offers).
- Decision-Making Capabilities: These involve the evaluation criteria and strategies agents employ to maximize objectives, such as utility functions or preference hierarchies. In the context of configuration systems, this relates to formulating and adapting configurations.
- Negotiation Process Dynamics: This aspect examines the overarching models and participant behaviors in negotiation scenarios, enabling the integration of behavioral tendencies linked to configuration commitments.

wo critical challenges must be addressed to enable multi-agent negotiation: defining mechanisms for communication and generating meaningful content for negotiation exchanges. The former is governed by the negotiation protocol, which establishes the rules and sequence of actions during the negotiation. The latter relies on the agent's decision-making model, which determines the rationale behind the proposals and responses. Negotiation protocols specify permissible actions, the timing of these actions, and the conditions under which agreements are finalized. For instance, the Alternating Multiple Offers Protocol (AMOP) emphasizes continuous proposal generation and voting, facilitating equitable participation and balanced outcomes (Baarslag, 2014). Decision-making models, on the other hand, focus on the reasoning processes agents use to evaluate options and optimize their outcomes. These models, when integrated with a flexible negotiation protocol, are crucial in determining the success of an agent's strategy (Baarslag et al., 2018). In our case, the negotiation domain corresponds to the configuration space derived from the configurators' experience. Agents' preferences, informed by their configuration commitments, guide their decisions within this domain. Proposals that fail to meet a minimum utility threshold, as determined by the agents' adaptive behavior, are iteratively refined to enhance their acceptability.

Software tools such as GENIUS and NegMAS have been developed to facilitate multi-agent negotiation. These platforms provide programmatic resources and frameworks for defining negotiation domains, preference profiles, and protocols. GENIUS (Hindriks et al., 2009) , implemented in Java, supports bilateral negotiation research by offering tools for configuring negotiation sessions, simulating interactions, and analyzing outcomes through graphical and statistical summaries. Complementary to GENIUS is Jupiter, a Python-based extension that integrates machine learning libraries, enhancing agents' ability to participate in GENIUS-managed negotiations (Fukui & Ito, 2018). NegMAS (Mohammad et al., 2021) , Python-based,

is designed for dynamic negotiation environments, supporting interdependent negotiations and agents with variable utility functions and flexible mechanisms beyond conventional protocols.

3.3 Architecture and implementation

The analysis of the proposed scenario delineates the abstract elements that compose the system architecture, as illustrated in Figure 5. From this framework, two distinct phases emerge, each serving a specific purpose: the preparation phase and the configuration phase. These phases are examined in detail below.

The preparation phase simulates professional training experiences by generating experimental performance data. This data enables agents to evaluate and propose configurations based on their understanding of prediction techniques in diverse work scenarios. The agents employ a utility calculation mechanism to quantify the suitability of potential configurations and implement a negotiation strategy based on weighted distances to facilitate consensus. Work scenarios are characterized by variables such as dataset properties, algorithm types and features, and configuration options.

Figure 5 illustrates the interplay of key entities in this phase. The seed datasets entity contains datasets with controlled variations in attributes, such as the number of records, class distributions, and variable selection. The algorithm descriptors entity provides information about algorithms, including their learning paradigm, generalization capabilities, and preprocessing requirements. The configuration options entity outlines the modular components of algorithm configurations and potential parameter settings. Insights from studies such as Villarreal's (Villarreal-Hernández et al., 2023), focusing on the KNN algorithm family, inform the structure of these options. For instance, configuring the KNN algorithm involves decisions about data preparation, heuristic selection, and parameter optimization, such as the k-value.

The experiment generator integrates seed datasets with configuration options to perform cross-validation experiments. This process generates performance data by iterating over compatible configurations, applying transformations to datasets (e.g., modifying columns or records), and expanding the range of experimental scenarios. The resulting experience corpus consolidates this performance data, associating it with descriptors related to dataset characteristics, algorithm configurations, and prediction problems. For the KNN algorithm family, the experience dataset includes 18 fields grouped into descriptors, as detailed in Table 1. This dataset serves as a repository of algorithm behavior under varying conditions, providing agents with examples that link performance outcomes to specific configurations and scenarios.

Table 1. Characteristics of the experiments, grouped by descriptor group

Descriptor group	Experiment characteristic field
Prediction problem descriptors	- Classes to consider - Target class - Variable preselection - Present categorical fields
Dataset descriptors	- Number of records - Present classes - Size difference among classes - Number of variables
Algorithm descriptors	- Learning type - Association type - Requires modification of the dataset - Produces a generalization model
Algorithm configuration descriptors	- Closeness metric - K parameter value - Class assignment policy - Dataset record reduction
Performance descriptors	- Mean precision - Standard deviation of precision

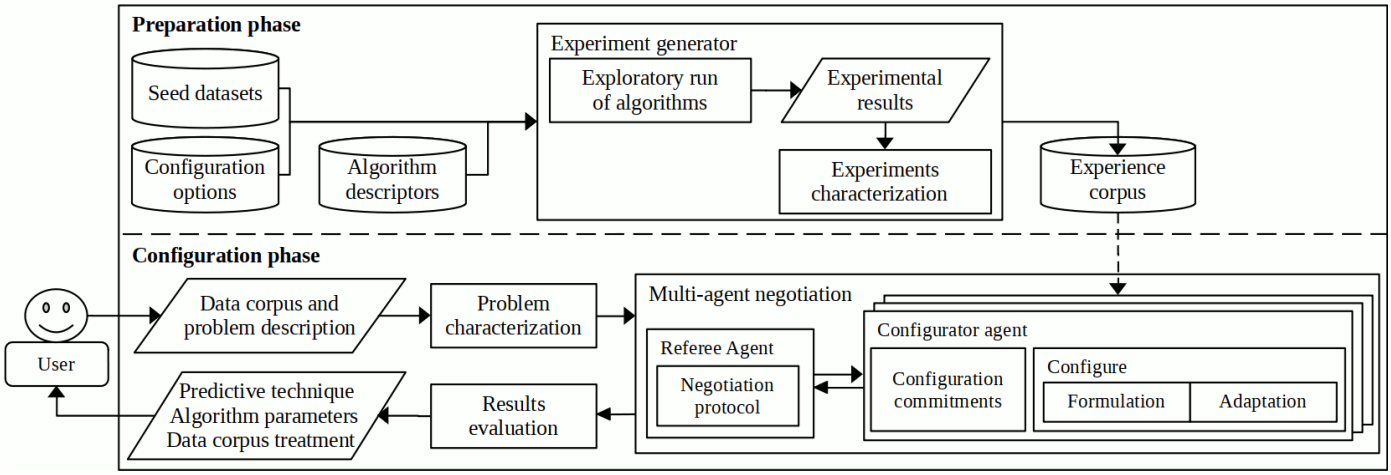

Fig. 4. The architecture for parameter configuration, including modules as negotiating agents

The configuration phase focuses on achieving consensus among configurators to determine the optimal configuration for a given prediction task. Negotiation interactions enable agents to exchange proposals and converge on solutions that satisfy their respective configuration commitments. The negotiation strategy seeks to identify configurations that perform well in scenarios analogous to the query while aligning with individual commitments.

This phase begins with user interaction. The user provides a dataset and defines the prediction problem through a simplified from that captures key descripts (Table 1). The user specifies elements such as the classes of interest, prior variable selection procedures, and dataset composition (e.g., numeric or categorical data). Dataset characteristics, such as record count, class distribution, and variable count, can be obtained through direct analysis. These steps are encapsulated in the problem characterization entity (Figure 5), which produces a query string aligned with the experience corpus.

The query is then passed to the multi-agent negotiation system. The referee agent manages the negotiation session, initializing the process by communicating the prediction problem to the configurator agents. During this preamble, agents prepare their reasoning frameworks and negotiation strategies.

A novel negotiation strategy based on lazy learning and weighted distance metrics is proposed for the configurator agents. This approach mirrors human problem-solving, which relies on identifying similar cases from past experiences. Two human traits—experience and preferences—affect the perception of similarity. In this system, the experience is shared among agents through the experience dataset, while preferences capture individual perspectives on the problem.

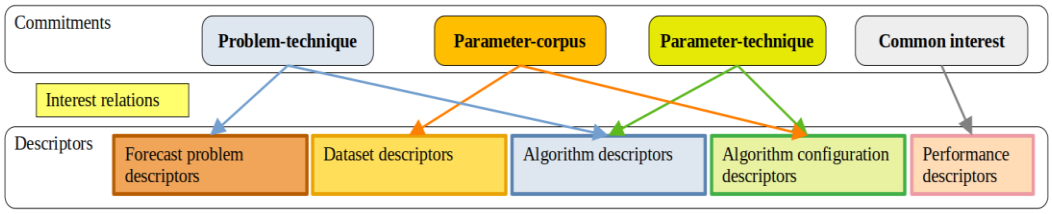

Fig. 5. Interest relationships between commitments and descriptors.

The preparation of preference profiles and weight vectors was grounded in a reasoning process that aligns with the commitments researchers adopt when addressing prediction tasks. These commitments inherently reflect a biased behavior, which can be systematically modeled through a preference scheme combined with the application of weighted distance metrics. To formalize the preferences of the agents, the experience dataset generated during the preparation phase serves as the foundational reference.

Each descriptor within the experience dataset is treated as a distinct set of variables, and the experiments encapsulated in this dataset are represented as records described by these variables. These records constitute a unique collection of objects that must be evaluated through preference mechanisms. Interest relationships are established between the descriptors in the experience dataset and the commitments each agent embodies. Figure 6 illustrates these relationships, wherein each configurator agent exhibits a differentiated level of interest in the descriptors, contingent upon their specific commitment.

Interest relationships are categorized into two levels: high and medium. While most descriptors are assigned medium interest by default, those directly aligned with an agent's commitment—including algorithm evaluation—are classified as having high interest. For instance, agents committed to problem-technique analysis prioritize prediction problem descriptors, algorithm descriptors, and algorithm evaluation descriptors. The magnitude of differentiation between high and medium interest is governed by a differentiation rate, which quantifies the disparity in importance. These interest relationships fulfill two critical objectives:

1. Utility Impact Definition: They influence the contribution of each descriptor to the calculation of utility for a given parametric configuration. Drawing on methods for calculating utility via normalized preference profiles (Baarslag et al., 2018; Hindriks et al., 2009; Lin et al., 2014), preference values are assigned to negotiation-relevant topics, corresponding to the descriptors. Descriptors classified as high interest exert a greater influence on the utility computation than those with medium interest, although both contribute to the aggregate utility score.
2. Dimensionality Reduction for Distance Calculation: They refine the experimental characterization space by minimizing the influence of less relevant descriptors on distance metrics. Consequently, when comparing a user-provided case with the experience dataset, agents perceive experiments with relevant interest relationships as being closer in reduced-dimensionality spaces.

When tasked with resolving a prediction problem, agents identify the most analogous case within their experience dataset. This involves calculating the distance between the new case and historical experiments, with utility and distance demonstrating an inverse relationship. Configurations that are closer in this reduced space exhibit higher utility values.

Given that utility and distance are computed relative to interest relationships, agents' rankings of configurations reflect their unique commitments. As a result, even when agents share access to the same experience dataset, their prioritized configurations may differ. Each agent independently ranks configurations by utility, with the top-ranked configuration—characterized by maximum utility and minimum distance—representing the ideal offer for the corresponding configuration commitment.

The acceptance threshold represents a tolerance level that allows for a controlled trade-off between maximizing utility and achieving a faster agreement. Initially, this threshold is set to a strict value to prioritize high-utility configurations. A proposal from one agent is deemed acceptable by another if the perceived utility of the proposal meets or exceeds the accepting agent's threshold. Ideally, all agents propose their optimal configurations in the initial round, and a unanimous agreement is reached. However, if consensus is not achieved, agents can iteratively adjust their acceptance thresholds to facilitate an agreement while minimizing utility loss. The extent of these adjustments is determined by the accumulated negative interactions from prior negotiation rounds. A negative interaction occurs when a configuration proposed by one agent is rejected through a vote by another.

In this context, the list of alternative offers comprises configurations from the experience dataset that meet the updated acceptance thresholds. Since this dataset has been pre-analyzed and ordered, alternative configurations are readily available for evaluation. With these negotiation elements defined, the referee agent oversees the execution of the negotiation protocol.

During the configuration phase, the referee agent implements and manages the negotiation protocol. For this architecture, the Alternating Multiple Offerings Protocol (AMOP) (Baarslag, 2014) was selected due to its ability to ensure equitable participation among agents and to incorporate inter-agent feedback. AMOP operates in iterative rounds, where each round is divided into an offer phase and a voting phase:

- Offer Phase: The referee agent prompts each configurator agent to generate and submit an offer, defined as a proposed configuration for the prediction task.
- Voting Phase: The referee agent sequentially presents these offers to all other configurator agents, soliciting their votes for each proposal. Each agent evaluates the proposals based on their own preference profiles and communicates their votes to the referee agent, who aggregates and reports the results.

f unanimous agreement is reached, the negotiation concludes. Otherwise, additional rounds are initiated, with agents refining their proposals based on the negotiation dynamics.

In the initial offer phase of the first round, agents propose their ideal offer—the configuration with the highest utility derived from the entire experience dataset. In subsequent rounds, agents incorporate the offers of others into their decision-making. Specifically, each agent identifies the most favorable offer from other agents (based on its own preference profile) and selects a

configuration from its alternative offers that minimizes the average distance to these favorable offers. This iterative process balances exploration of alternative solutions with adherence to individual preferences, thereby increasing the likelihood of reaching consensus.

Upon achieving an agreement—or exhausting the predefined round limit—the referee agent proceeds to execute the selected configurations using the provided working dataset. This process involves two steps: (1) applying any required modifications to the dataset and (2) generating an instance of the chosen algorithm configured with the negotiated parameters. Cross-validation is performed on this instance to evaluate its performance, and the results are presented to the user. This output includes a ranked list of configurations and the corresponding data necessary for the user to execute their prediction task with unclassified data.

The proposed architecture exhibits a high degree of scalability and adaptability. The three storage entities utilized by the experiment generator—seed datasets, algorithm descriptors, and configuration options—can be incrementally expanded as new research on algorithms and datasets becomes available. The experiment generator employs a greedy exploration strategy, iterating through combinations of options to produce a cumulative experience corpus. Additionally, this corpus can be enriched by mining performance data from existing publications, provided the experiments are reproducible or their results can be directly integrated into the corpus.

During the configuration phase, the system's flexibility is further enhanced by the use of interest relationships. These relationships enable the integration of new configuration commitments and descriptors, thereby enriching the negotiation process. Importantly, the negotiation strategy is designed to operate dynamically on the current experience corpus, eliminating the need for recalculating generalization models. This ensures that the architecture continuously incorporates the most up-to-date knowledge.

As the number of configuration commitments increases, so does the number of configurator agents involved in the negotiation. Despite this, the computational overhead remains manageable. Each configurator agent performs a lightweight analysis of the experience corpus during the preamble phase of negotiation, ensuring rapid execution.

The architecture was implemented using the Python programming language, taking advantage of the facilities offered by the NumPy (Harris et al., 2020) and Scikit-learn (Pedregosa et al., 2011) libraries.

4 Proof of concept

4.1 Setting up the preparation phase

The proof of concept was conducted to validate the proposed architecture, beginning with the configuration of the storage entities feeding the experiment generator. During the preparation phase, the seed datasets utilized were classical benchmark datasets commonly employed in academic classification studies: Wine, Iris, and Breast Cancer. The case study selected for this proof of concept was based on the work of Loredo (Loredo-Pong et al., 2022) , where the configuration options and algorithm descriptors focused on the K-Nearest Neighbors (KNN) algorithm. These descriptors and options were adapted from (Villarreal-Hernández et al., 2023) and are summarized in Table 2.

For the estimation of the *K*-value, the parameter *n* represents the number of records in the dataset under configuration. The experiment generator utilized the seed datasets to produce modified datasets by applying a combination of variable estimation and record reduction options. This process resulted in a total of 36 datasets available for experimentation. The generator conducted cross-validation by combining configurations from the various option categories. For *K*-values, the generator identified the highest value produced by the options and generated a range of experiments with $K \in [1,\max(K)]$. Inoperative or redundant configurations, such as cases where $K=1$ combined with frequency count, were discarded, as these settings behave identically to direct assignment. Instead, the minimum *K* value allowed for frequency count policy is 3. As shown in Table 2, the current repertoire of distance functions does not include options specifically designed to handle categorical label values directly. The distance functions were chosen around the case study, this does not require addressing categorical fields, therefore, this delimitation has no impact on performance.

Each configuration variant and modified dataset produced a distinct experiment. A total of 3,359 experiments were executed over a runtime of 24:05 minutes, and the results were stored in the experience corpus. The evaluation of configuration precision

was conducted using the balanced accuracy metric, which emphasizes a more equitable assessment across all classes. Unlike standard accuracy, which can be skewed by the performance on majority classes, balanced accuracy ensures fairness by averaging the recall for each class, thereby providing a more representative measure of overall performance (Brodersen et al., 2010; Kelleher et al., 2015).

Table 2. Configuration options implemented for the proof of concept around the KNN algorithm are listed by category. In the estimation of K value column *n* is the number of records in the dataset

Dataset variable estimation	Dataset record reduction	Closeness-similarity metrics	Estimation of K value	Class assignment policies
None	None	Euclidean	7 (Recommended from literature)	Frequency count
Pearson	Centroids	Manhattan	1 (Required by Nearest Neighbor)	Direct assignment
Chi-square	Fast condensation (Angiulli, 2005)	Minkowski p=1.5	The square root of *n*	Simple majority
One hot encoder		Minkowski p=100	$n^{2/8}$ (Enas & Choi, 1986)	Qualified majority
		Chebyshev	$n^{3/8}$ (Enas & Choi, 1986)	Neighbors with weights

4.2 Setting up the configuration phase

During the forecast phase, the configurable parameters pertained to the multi-agent negotiation process. The referee agent was configured with a maximum limit of 200 negotiation rounds. Additionally, the roles for the configurator agents—problem-technique, parameter-corpus, and parameter-technique—were defined and assigned. Once invoked, each configurator agent initialized its configuration preamble, during which preferences and weights for the negotiation strategy were generated. These values were dependent on the differentiation rate, which was set to 0.15 in this proof of concept.

The case study by Loredo (Loredo-Pong et al., 2022) employed the KNN algorithm to predict the state of aggregation for substances, classifying them into four categories: solution (S), gel (G), precipitate (P), and insoluble (I). The study analyzed a series of alkoxybenzoates and solvents, characterized by their Hansen Solubility Parameters and the number of carbons in the alkyl tail. During the development of this work, several training datasets were generated, as detailed in Table 3. The "Data Type" column in Table 3 indicates from left to right the composition of the dataset with the form Type (consecutive columns of the same type).

Table 3. Characteristics of the datasets generated by Loredo. The aggregation states were obtained from experiments in the laboratory

Dataset	Variables	Population	Distribution by classes (G, I, P S)	Data Type
A	38	240	28, 4, 7, 203	Int(1), Binary(30), Int(7)
B	24	270	47, 3, 3, 217	Binary(15), Decimal(1), Int(4), Decimal(3), Int(1)
C	11	270	47, 3, 3, 217	Int(1), Decimal(3), Int(3), Decimal(3), Int(1)
D	24	270	47, 223	Binary(15), Decimal(1), Int(4), Decimal(3), Int(1)
E	11	270	47, 223	Int(1), Decimal(3), Int(3), Decimal(3), Int(1)

Datasets D and E were transformed into binary classification problems by grouping classes into gel and non-gel categories. All datasets contained numerical representations of physicochemical descriptors (e.g., Ester, Ether, Polarity) and, in the case of

datasets A, B and D, included processed categorical labels. These categorical variables were processed using One-Hot Encoding, as shown in the configurations LrdA, LrdB, and LrdD in Table 4.

In Figure 6, the elements discussed in Sections 4.1 and 4.2 are highlighted, illustrating their relationship with the entities that comprise the configuration architecture.

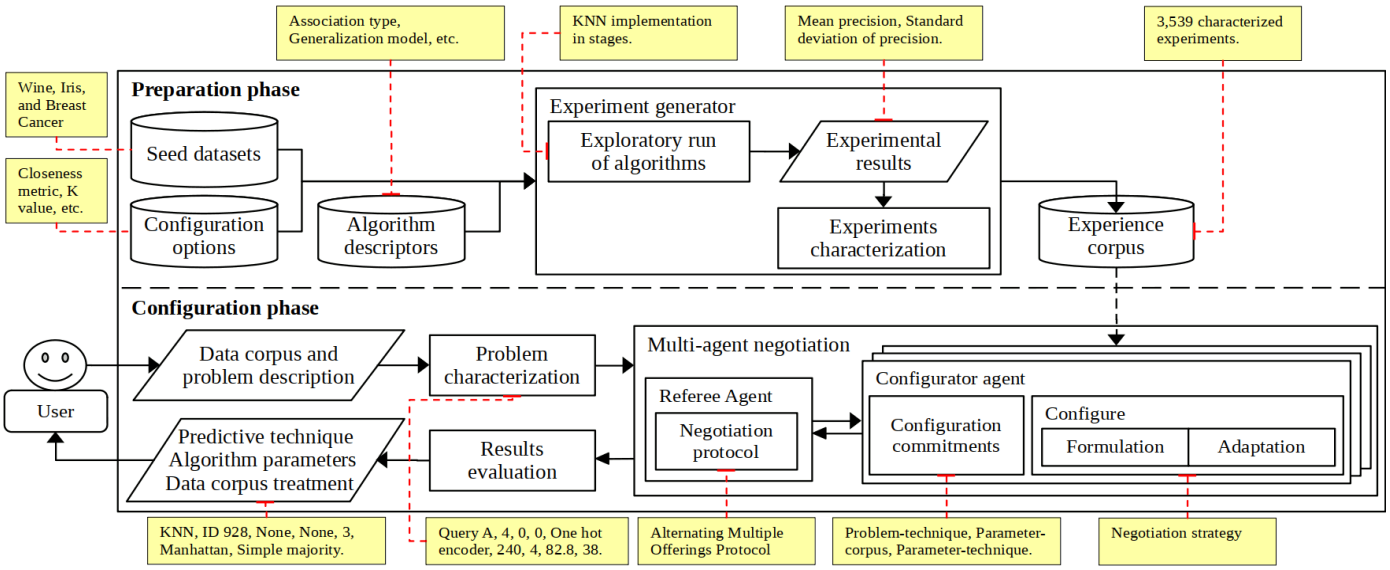

Fig. 6. The architecture for parameter configuration, the annotations indicate the elements discussed in this section}

4.3 Experimentation with KNN configurations

Using each training set, Loredo evaluated various configurations of the KNN algorithm. These configurations, identified in Table 4 as Lrd+Dataset+Configuration Number, were defined according to the order and conventions presented in (Villarreal-Hernández et al., 2023). For the class assignment policy, the inverse of the distance was employed as the weighting schema for neighbors.

Table 4. Configurations proposed by Loredo (named Lrd+Dataset+Configuration Number) and proposed by the configurator agents (named Id+Configuration Number) for the different datasets of the gelation prediction problem

Algorithm configuration name	Variable estimation	Record reduction	K	Distance metric	Class assignment policy	Dataset
ID 928	None	None	3	Manhattan	Simple majority	A
LrdA1	One hot encoder	None	5	Euclidean	Frequency count	A
LrdA2	One hot encoder	None	10	Euclidean	Neighbors with weights	A
ID 1633	None	None	3	Manhattan	Simple majority	B
LrdB1	One hot encoder	None	3	Euclidean	Frequency count	B
LrdB2	One hot encoder	None	3	Euclidean	Neighbors with weights	B
LrdB3	One hot encoder	None	5	Euclidean	Neighbors with weights	B
LrdB4	One hot encoder	None	10	Euclidean	Neighbors with weights	B
ID 851	One hot	Centroids	1	Manhattan	Direct assignment	C

	encoder					
ID 1791	None	Centroids	1	Manhattan	Direct assignment	C
LrdC1	None	None	3	Euclidean	Frequency count	C
LrdC2	None	None	5	Euclidean	Frequency count	C
LrdC3	None	None	5	Euclidean	Neighbors with weights	C
LrdC4	None	None	10	Euclidean	Neighbors with weights	C
ID 1633	None	None	3	Manhattan	Simple majority	D
LrdD1	One hot encoder	None	5	Euclidean	Neighbors with weights	D
LrdD2	One hot encoder	None	10	Euclidean	Neighbors with weights	D
ID 1633	None	None	3	Manhattan	Simple majority	E
LrdE1	None	None	3	Euclidean	Neighbors with weights	E

Table 5 presents the user queries formulated to resolve the configurations of each dataset listed in Table 3. The first column of Table 5 specifies the query field representing the forecast situation, while the subsequent columns display the corresponding feature values for each dataset query. These columns describe the system's input query parameters in detail, which serve as the basis for configuration resolution.

Table 5. Queries to the configuration system with their characteristics, one string per column. A query is entered for dataset composition associated with their letter

Characteristic field	Query for dataset A	Query for dataset B	Query for dataset C	Query for dataset D	Query for dataset E
Classes to consider	4	4	4	2	2
Present categorical fields	0 (None)	0 (None)	0 (None)	0 (None)	0 (None)
Target class	0 (None)	0 (None)	0 (None)	0 (None)	0 (None)
Variable preselection	One hot encoder	One hot encoder	None	One hot encoder	None
Number of records	240	270	270	270	270
Present classes	4	4	4	2	2
The size difference between classes	82.8	88.2	88	88	88
Number of variables	38	24	11	24	11

The outcomes generated by the configurator agents for each query are summarized in Table 6. For Dataset A, the configurators reached an agreement on example 928 from the experience corpus. Similarly, agreements were achieved for examples 1633 in

contexts B, D, and E, and for examples 851 and 1791 in context C. It is important to note that the information returned by the agents originates from the experience corpus, which may include data not directly relevant to the end user.

The precision measures reported in Table 6 correspond to the results recorded during experimentation and do not necessarily reflect the performance of the final solution. For instance, the average accuracy associated with example ID 851 is 0, likely due to the combined effects of using a nearest neighbor approach (K=1) and a direct assignment policy. These setting are known for their instability, particularly when paired with variable estimation techniques such as One-Hot Encoding, which may introduce additional classification errors.

The configurations proposed by the configurator agents, along with their corresponding dataset applications, are detailed in Table 4. The variable estimation option is applied cumulatively; that is, if a previous variable estimation technique has been applied, the new one is incorporated as well. A "None" value for variable estimation indicates that no additional technique is applied.

Table 6. Output chain of the configuration system with its characteristics, one configuration per column. The relation of the configurations is the following: for query A, configuration ID 928; B, D, and E, ID 1633; C, ID 851 and 1791

Characteristic field	ID 928	ID 1633	ID 851	ID 1791
Learning type	Example-based	Example-based	Example-based	Example-based
Requires modification of the dataset	0 (No)	0 (No)	0 (No)	0 (No)
Association type	Proximity	Proximity	Proximity	Proximity
Produces a generalization model	0 (No)	0 (No)	0 (No)	0 (No)
Dataset record reduction	None	None	None	None
Dataset variable estimation	None	None	One hot encoder	One hot encoder
Closeness metric	Manhattan	Manhattan	Manhattan	Manhattan
K parameter value	3	3	1	1
Class assignment policy	Simple majority	Simple majority	Direct assignment	Direct assignment
Mean precision	0.796078431372549	1	0	0.00392156862745098
Standard deviation of precision	0.0363671313548851	0	0	0.00784313725490196

The returned configurations were evaluated using a 5-fold cross-validation scheme. Additionally, the configurations reported by Loredo were reproduced for comparative purposes. Across all executed configurations, three key performance metrics were assessed: Precision (shown in Table 7), Dispersion (standard deviation, shown in Table 8), Execution Time in seconds (shown in Table 9).

These performance metrics were statistically analyzed using the Friedman rank test (Friedman, 1937) and then Holm test (Holm, 1979). García (García et al., 2010) provide a comprehensive analysis of statistical tests for comparing multiple classifies across multiple datasets, reinforcing the validity of using the Friedman test under conditions similar to those of our study. This methodology ensures that the assumptions of the Friedman test are respected, particularly when applied to the context of classifier comparisons with repeated measurements. We believe this aligns with the insights provided in (Demsar, s.f.), which emphasizes the importance of meeting these conditions.

Following the recommendations of García, both the classic Friedman test and the aligned ranks Friedman test were employed, depending on the number of configurations compared: for datasets with more than four configurations (contexts B and C), the classic Friedman test was applied; for datasets with four or fewer configurations (contexts A, D, and E), the aligned ranks Friedman test was used. The ranks derived from these tests were subsequently processed with the Holm test to determine statistical significance.

The results of the statistical analysis indicate that the Friedman test detected significant differences among the configurations' performances, ruling out ties in the rankings. Furthermore, the rejection of the null hypothesis (H_0) in the Holm test confirms that these differences are statistically significant.

Tables 7, 8, and 9 provide the detailed results of these analyses, grouped by datasets A to E, and include the performance metrics along with their respective Friedman ranks and Holm test results. For the implementation of the statistical tests, the STAC Python library (Rodriguez-Fdez et al., 2015) was employed. For precision metrics, a higher rank is considered indicative of better performance. Conversely, for dispersion and execution time, lower ranks denote superior outcomes, as these metrics benefit from minimization.

Table 7. Precision configuration evaluation, grouped by dataset. In each group, the configurations are ordered according to their Friedman rank. Columns marked with Holm comparison H_0 (column five onwards) present the VS between configurations

Dataset	Algorithm	Mean precision	Friedman rank	Holm comparison H_0					
				ID 928	LrdA1	LrdA2			
A	ID 928	88.00%	2.6	--	Reject	Reject			
	LrdA1	87.65%	2.4	Reject	--	Accept			
	LrdA2	60.48%	1	Reject	Accept	--			
				ID 1633	LrdB1	LrdB2	LrdB3	LrdB4	
B	ID 1633	85.70%	4.6	--	Reject	Reject	Reject	Reject	
	LrdB1	85.50%	4.4	Reject	--	Reject	Reject	Reject	
	LrdB2	71.11%	3	Reject	Reject	--	Reject	Accept	
	LrdB3	63.69%	2	Reject	Reject	Reject	--	Reject	
	LrdB4	50.18%	1	Reject	Reject	Accept	Reject	--	
				ID 851	ID 1791	LrdC1	LrdC2	LrdC3	LrdC4
C	LrdC1	84.00%	5.45	Reject	Reject	--	Reject	Reject	Reject
	ID 1791	83.73%	5.34	Reject	--	Reject	Reject	Accept	Reject
	LrdC2	81.80%	3.92	Reject	Reject	Reject	--	Accept	Reject
	ID 851	80.59%	3.29	--	Reject	Reject	Reject	Reject	Accept
	LrdC3	60.39%	2	Reject	Accept	Reject	Accept	--	Reject
	LrdC4	47.72%	1	Accept	Reject	Reject	Reject	Reject	--

				ID 1633	LrdD1	LrdD2
D	ID 1633	88.00%	3	--	Reject	Reject
	LrdD1	74.23%	2	Reject	--	Reject
	LrdD2	63.14%	1	Reject	Reject	--
				ID 1633	LrdE1	
E	ID 1633	87.00%	2	--	Reject	
	LrdE1	75.94%	1	Reject	--	

For Dataset A, the agents' proposed configuration demonstrated a slight improvement in precision and deviation (see Table 8) compared to previous configurations. However, it exhibited the highest computational cost in terms of execution time, as shown in Table 9.

In the case of Dataset B, the agents' configuration outperformed previous proposals in terms of precision (see Table 7) and execution time (see Table 9), securing a close second place in terms of deviation.

For Dataset C, the agents' configuration ID 1791 achieved the highest precision and deviation ratings, albeit with a small margin to the second-best configuration. In contrast, configuration ID 851 was the fastest, but this came at the cost of lower precision and higher deviation, highlighting its limitations in predictive reliability.

In Datasets D and E, the differences in precision among configurations were more pronounced compared to other contexts. Configuration ID 1633 provided a significant improvement in precision over previous proposals while incurring only a minor increase in computational time (see Table 9).

Across all datasets, the agents' configurations outperformed previous proposals in precision in four out of five cases (contexts A, B, D, and E) as shown in Table 7. However, this improvement was not consistently accompanied by a reduction in deviation, as evident in contexts B and C.

Table 8. Standard deviation of precision configuration evaluation, grouped by dataset. In each group, the configurations are ordered according to their Friedman rank. Columns marked with Holm comparison H_0 (column five onwards) present the VS between configurations

Dataset	Algorithm	Standard deviation	Friedman rank	Holm comparison H_0					
				ID 928	LrdA1	LrdA2			
A	ID 928	0.0342	1.48	--	Reject	Reject			
	LrdA1	0.0368	1.65	Reject	--	Accept			
	LrdA2	0.0660	2.87	Reject	Accept	--			
				ID 1633	LrdB1	LrdB2	LrdB3	LrdB4	
B	LrdB1	0.0448	2.48	Reject	--	Accept	Accept	Accept	
	ID 1633	0.0453	2.58	--	Reject	Accept	Accept	Accept	
	LrdB2	0.0510	2.77	Accept	Accept	--	Accept	Accept	
	LrdB3	0.0609	3.48	Accept	Accept	Accept	--	Accept	
	LrdB4	0.0635	3.68	Accept	Accept	Accept	Accept	--	
				ID 851	ID 1791	LrdC1	LrdC2	LrdC3	LrdC4

C	LrdC1	0.0387	2.45	Reject	Accept	--	Accept	Accept	Reject
	ID 1791	0.0393	2.71	Reject	--	Accept	Accept	Accept	Reject
	ID 851	0.0399	2.9	--	Reject	Reject	Reject	Reject	Accept
	LrdC2	0.0443	3.16	Reject	Accept	Accept	--	Accept	Reject
	LrdC4	0.0602	4.77	Accept	Reject	Reject	Reject	Reject	--
	LrdC3	0.0654	5	Reject	Accept	Accept	Accept	--	Reject
				ID 1633	LrdD1	LrdD2			
D	ID 1633	0.0344	1.61	--	Accept	Accept			
	LrdD1	0.0491	2	Accept	--	Accept			
	LrdD2	0.0554	2.39	Accept	Accept	--			
				ID 1633	LrdE1				
E	ID 1633	0.0407	1.4	--	Reject				
	LrdE1	0.0458	1.6	Reject	--				

Examining the agents' configurations across contexts reveals recurring patterns. In contexts A, B, D, and E, the agents' configurations did not include variable estimation or record reduction procedures (see Table 4). This suggests that the natural structure of the data for these specific problems is sufficient, and preprocessing techniques may introduce unnecessary complexity or degrade model performance. In all four contexts, the agents consistently selected K=3, an odd value recommended in the literature to avoid ties during class assignment. The relationship between K and the class assignment policy is particularly significant. Using the simple majority policy, at least two out of three neighbors must belong to the same class for a successful classification. This strategy is especially effective when the dataset excludes at least one class, as observed in Datasets A and B, simplifying the classification task. For Datasets D and E, using K=3 mitigates the sensitivity to noise, a common issue when relying on a single nearest neighbor (K=1).

Table 9. Time in seconds configuration evaluation, grouped by dataset. In each group, the configurations are ordered according to their Friedman rank. Columns marked with Holm comparison H_0 (column five onwards) present the VS between configurations

Dataset	Algorithm	Time in seconds	Friedman rank	Holm comparison H_0					
				ID 928	LrdA1	LrdA2			
A	LrdA1	1.8741	1.68	Reject	--	Accept			
	LrdA2	1.8800	1.77	Reject	Accept	--			
	ID 928	1.9097	2.55	--	Reject	Reject			
				ID 1633	LrdB1	LrdB2	LrdB3	LrdB4	
B	ID 1633	2.3339	2.77	--	Accept	Accept	Accept	Accept	
	LrdB2	2.3366	2.87	Accept	Accept	Accept	Accept	--	
	LrdB4	2.3398	2.9	Accept	Accept	--	Accept	Accept	
	LrdB3	2.3351	2.97	Accept	Accept	Accept	--	Accept	
	LrdB1	2.3450	3.48	Accept	--	Accept	Accept	Accept	
				ID 851	ID 1791	LrdC1	LrdC2	LrdC3	LrdC4
C	ID 851	2.3358	2.32	--	Reject	Accept	Reject	Reject	Accept
	LrdC4	2.3786	2.35	Accept	Reject	Accept	Reject	Reject	--

	LrdC1	2.3962	3.52	Accept	Accept	--	Accept	Accept	Accept
	ID 1791	2.3919	4.23	Reject	--	Accept	Accept	Accept	Reject
	LrdC2	2.4100	4.29	Reject	Accept	Accept	--	Accept	Reject
	LrdC3	2.4266	4.29	Reject	Accept	Accept	Accept	--	Reject
				ID 1633	LrdD1	LrdD2			
D	LrdD2	2.2402	1.9	Accept	Accept	--			
	LrdD1	2.2473	1.94	Accept	--	Accept			
	ID 1633	2.2433	2.16	--	Accept	Accept			
				ID 1633	LrdE1				
E	LrdE1	2.4975	1.45	Accept	--				
	ID 1633	2.5078	1.55	--	Accept				

Dataset C presented a unique challenge, where the agents' configurations yielded the lowest ratings compared to other contexts. Two notable differences were observed:

1. Multiple Configuration Proposals: unlike other contexts, agents proposed more than one configuration for the prediction task in Dataset C.
2. Inclusion of variable estimation and record reduction: unlike contexts A, B, D, and E, the agents included variable estimation and record reduction options in Dataset C. This involved the centroid reduction method combined with K=1 and the direct assignment policy.

Configuration ID 851, which utilized One-Hot Encoding for variable estimation, received the lowest precision and deviation scores. This result suggests that the One-Hot Encoding technique may not be suitable for Dataset C. Upon analyzing Dataset C's characteristics (see Table 5), it is evident that it contains four distinct classes and 11 variables, a reduction compared to other datasets. The poor performance of ID 851 appears to stem from its reliance on One-Hot Encoding, which is the main difference between it and the superior ID 1791 configuration.

The configurator agents demonstrated robust performance across most contexts, particularly when they avoided variable estimation and record reduction, allowing the natural structure of the data to guide predictions. The consistent selection of K=3 aligns with established literature and further validates its efficacy in reducing noise sensitivity and improving classification accuracy. However, Dataset C highlighted the limitations of including preprocessing techniques such as One-Hot Encoding, emphasizing the importance of tailoring configurations to the dataset's specific characteristics.

This analysis underscores the adaptability and effectiveness of the configurator agents, while revealing opportunities for refinement in cases where the dataset complexity or preprocessing steps may undermine predictive performance.

5 Discussion

The architecture for parameter configuration has been designed to facilitate the continuous incorporation of new datasets and algorithms. While integrating entire families of algorithms is optimal for conducting reuse studies, the proposed methodology for generating and storing experiential data supports the inclusion of isolated algorithms as well.

Dataset Selection Approaches

The selection of seed datasets represents a critical aspect of the architecture's capabilities and can be addressed through various methodologies. One approach involves leveraging datasets specific to the chemical domain, enabling the extraction of relevant descriptors that capture the strategies historically used to solve similar problems. Alternatively, datasets from other scientific domains may be employed, provided their descriptors are sufficiently diverse to broaden the system's perspective and enrich its parametric configuration capabilities. A further method entails the use of synthetic dataset generation techniques, which may be linked to real-world problems or entirely artificial. This approach offers the dual advantages of statistical control over dataset composition and the opportunity to explore parameter configurations under systematically defined conditions.

Capturing User Queries
A significant challenge lies in capturing end-user queries, as formulating a prediction problem in logical and computable terms is inherently complex. The definition of the prediction task can directly influence the difficulty of identifying optimal parametric configurations. This is because predictive algorithms are associated with computational complexities that often depend on their parameter settings, which, in turn, may include elements derived from the problem description.

To address this, the architecture can incorporate a wizard interface that assists users in constructing a simplified representation of their prediction problem. This interface ensures that the problem remains compatible with the configurator agents, avoiding scenarios where overly exotic problem definitions impede effective parameter configuration.

Adaptability of the Multi-Agent Negotiation Framework
The multi-agent negotiation framework within the architecture is designed to accommodate the incorporation of new knowledge dynamically. Adding examples to the experience corpus requires only the extraction of their descriptors, eliminating the strict dependency on the experiment generator for data generation.

The preference-generation method ensures that for any input within the experience corpus, it is possible to compute the utility of each commitment. Furthermore, the interest relationships schema introduces a layer of abstraction between commitments and specific descriptor variables. This abstraction facilitates: modifications to descriptor variables without affecting the overall structure; and addition of new descriptors or commitments stemming from subsequent studies. This design ensures the architecture remains flexible and scalable, capable of integrating new descriptors, algorithms, and datasets as the field evolves.

6 Conclusions and Future Work

The proposed architecture facilitates the exploration of diverse configuration options for predictive algorithms. By leveraging the multi-agent paradigm, it enables the evaluation of algorithmic configurations from multiple perspectives, achieving a balanced assessment through consensus mechanisms. Data mining and machine learning algorithms have demonstrated remarkable success in predicting data trends and have been applied extensively to solve complex challenges in chemical and material processes (Goh et al., 2017). Expanding the capabilities of the configuration agents within this framework promises faster and more efficient deployment of these computational techniques.

This architecture is built upon a lazy learning paradigm that supports the incremental inclusion of algorithmic options and dataset types, thereby enhancing the cumulative experience of the agents. While the architecture incorporates an experiment generator that requires algorithm implementation, the experience corpus can be enriched with algorithmic configurations derived from existing literature. The inclusion of additional agent roles can be achieved by defining new interest relationships, commitments, or descriptors, thereby broadening the scope of configuration strategies. Experimental variations may include generating experience corpora using alternative seed datasets to diversify the agents' perspectives. Furthermore, adjustments to the differentiation rate—a parameter reflecting the magnitude of interest differences among variables—can influence the rigidity of negotiation strategies, potentially altering consensus outcomes.

Despite the variety of techniques and methodologies addressing parameter optimization challenges, a review of the state-of-the-art reveals a lack of applications of the intelligent agent paradigm to this problem within the context of computational chemistry. This positions the integration of intelligent agent methodologies as a novel approach in this domain.

The application of parameter optimization to computational chemistry techniques has the potential to elucidate the relationships between algorithmic parameters and the investigated chemical phenomena. Understanding these relationships can lead to significant improvements in the accuracy of classifications and predictions, as well as reductions in energy consumption and time requirements in chemical research and experimentation.

7 Acknowledgments

JAVH gratefully acknowledges the CONAHCyT grant for doctoral studies (Grant 794328). The authors are also thankful for the support obtained from: Tecnológico Nacional de México through the Instituto Tecnológico de Ciudad Madero, for the use of

the Laboratorio Nacional de Tecnologías de Información (LaNTI), and for the support received through the TecNM project 22557.25-P Design of an intelligent virtual agent for decision support with impact on the navigation of electric vehicles.

References

Angiulli, F. (2005). Fast condensed nearest neighbor rule. In *Proceedings of the 22nd International Conference on Machine Learning – ICML '05* (pp. 25–32). https://doi.org/10.1145/1102351.1102355

Baarslag, T. (2014). *What to bid and when to stop* (Doctoral thesis). Technische Universiteit Delft.

Baarslag, T., Pasman, W., Hindriks, K., & Tykhonov, D. (2018). Using the Genius Framework for running autonomous negotiating parties. *In* T. Baarslag et al. (Eds.), *Proceedings of the 30th International Conference on Autonomous Agents and Multiagent Systems* (págs. 30).

Barca, G. M. J., Bertoni, C., Carrington, L., Datta, D., De Silva, N., Deustua, J. E., Fedorov, D. G., Gour, J. R., Gunina, A. O., Guidez, E., Harville, T., Irle, S., Ivanic, J., Kowalski, K., Leang, S. S., Li, H., Li, W., Lutz, J. J., Magoulas, I., Mato, J., Mironov, V., Nakata, H., Pham, B. Q., Piecuch, P., Poole, D., Pruitt, S. R., Rendell, A. P., Roskop, L. B., Ruedenberg, K., Sattasathuchana, T., Schmidt, M. W., Shen, J., Slipchenko, L., Sosonkina, M., Sundriyal, V., Tiwari, A., Galvez Vallejo, J. L., Westheimer, B., Włoch, M., Xu, P., Zahariev, F., & Gordon, M. S. (2020). Recent developments in the general atomic and molecular electronic structure system. *The Journal of Chemical Physics, 152*(15), 154102. https://doi.org/10.1063/5.0005188

Birattari, M. (2005). *The problem of tuning metaheuristics as seen from a machine learning perspective* (Doctoral dissertation). Université Libre de Bruxelles. https://iridia.ulb.ac.be/~mbiro/paperi/BirattariPhD.pdf

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition* (pp. 3121–3124). IEEE. https://doi.org/10.1109/ICPR.2010.764

Cayci, A., Eibe, S., Menasalvas, E., & Saygin, Y. (2011). Bayesian networks to predict data mining algorithm behavior in ubiquitous computing environments. In M. Atzmueller, A. Hotho, M. Strohmaier, & A. Chin (Eds.), *Analysis of Social Media and Ubiquitous Data* (LNCS, Vol. 6904, pp. 119–141). Springer. https://doi.org/10.1007/978-3-642-23599-3_7

Cerecedo-Córdoba, J. A., Barbosa, J. J. G., Terán-Villanueva, J. D., Frausto-Solís, J., & Flores, J. A. M. (2017). Use of neuroevolution to estimate the melting point of ionic liquids. *International Journal of Combinatorial Optimization Problems and Informatics, 8*(2), 2–9. Retrieved from https://ijcopi.org/ojs/article/view/8

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*(1), 1–30. Retrieved from http://www.jmlr.org/papers/v7/demsar06a.html

San Fabián, E. (2020). *Cálculos computacionales de estructuras moleculares* (Tesis de Máster). Universidad de Alicante.

Eiben, Á. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation, 3(2), 124–141.*

Enas, G. G., & Choi, S. C. (1986). Choice of the smoothing parameter and efficiency of k-nearest neighbor classification. *Computers & Mathematics with Applications, 12*(2), 235–244. https://doi.org/10.1016/0898-1221(86)90076-3

Fehri, H., Gooya, A., Lu, Y., Meijering, E., Johnston, S. A., & Frangi, A. F. (2019). Bayesian polytrees with learned deep features for multi-class cell segmentation. *IEEE Transactions on Image Processing, 28*(7), 3246–3260. https://doi.org/10.1109/TIP.2019.2895455

Fernandez, E., Rangel-Valdez, N., Cruz-Reyes, L., Gomez-Santillan, C., Rivera-Zarate, G., & Sanchez-Solis, P. (2019). Inferring parameters of a relational system of preferences from assignment examples using an evolutionary algorithm. *Technological and Economic Development of Economy, 25*(4), 693–715. https://doi.org/10.3846/tede.2019.9475

Finocchi, F. (2011). *Density functional theory for beginners* (Bachelor's thesis). Université Pierre et Marie Curie, Paris, France.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701. https://doi.org/10.1080/01621459.1937.10503522

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180*(10), 2044–2064. https://doi.org/10.1016/j.ins.2009.12.010

Goh, G. B., Hodas, N. O., & Vishnu, A. (2017). Deep learning for computational chemistry. *Journal of Computational Chemistry, 38*(16), 1291–1307. https://doi.org/10.1002/jcc.24764

Gordon, M. S., & Schmidt, M. W. (2005). Advances in electronic structure theory: GAMESS a decade later. In P. v. R. Schleyer (Ed.), *Theory and applications of computational chemistry* (pp. 1167–1189). Elsevier.

Gupta, J. K., Adams, D. J., & Berry, N. G. (2016). Will it gel? Successful computational prediction of peptide gelators using physicochemical properties and molecular fingerprints. *Chemical Science, 7*(7), 4713–4719. https://doi.org/10.1039/C6SC00722H

Hall, S. R., Allen, F. H., & Brown, I. D. (1991). The crystallographic information file (CIF): A new standard archive file for crystallography. *Acta Crystallographica Section A: Foundations of Crystallography, 47*(6), 655–685. https://doi.org/10.1107/S010876739101067X

Hanson, R. M. (2016). Jmol SMILES and Jmol SMARTS: Specifications and applications. *Journal of Cheminformatics, 8*, Article 50. https://doi.org/10.1186/s13321-016-0160-4

Li, H., Zhang, Z., & Zhao, Z.-Z. (2019). Data-mining for processes in chemistry, materials, and engineering. *Processes, 7*(3), Article 151. https://doi.org/10.3390/pr7030151

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hindriks, K., Jonker, C. M., Kraus, S., Lin, R., & Tykhonov, D. (2009). Genius: Negotiation environment for heterogeneous agents. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems* (Vol. 2, pp. 1425–1426). International Foundation for Autonomous Agents and Multiagent Systems.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics, 6*(2), 65–70.

Hutter, F., Kotthoff, L., & Vanschoren, J. (Eds.). (2019). *Automated machine learning: Methods, systems, challenges*. Springer. https://doi.org/10.1007/978-3-030-05318-5

Iglesias Fernández, C. Á. (1998). Fundamentos de los agentes inteligentes (pp. 7–40). Universidad Politécnica de Madrid.

Keith, J. A., Vassilev-Galindo, V., Cheng, B., Chmiela, S., Gastegger, M., Müller, K.-R., & Tkatchenko, A. (2021). Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chemical Reviews, 121*(16), 9816–9872. https://doi.org/10.1021/acs.chemrev.1c00107

Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics: Algorithms, worked examples, and case studies*. The MIT Press.

Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K., & Jonker, C. M. (2014). Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence, 30*(1), 48–70. https://doi.org/10.1111/j.1467-8640.2012.00463.x

Loredo-Pong, V., Morales-Rodríguez, M. L., Díaz-Zavala, N. P., Rangel-Valdez, N., & Sosa-Sevilla, J. E. (2022). A design and analysis of classification models for the gelification of alkoxybenzoates using the kNN algorithm. *International Journal of Combinatorial Optimization Problems and Informatics, 13*(2), 58–64. https://doi.org/10.61467/2007.1558.2022.v13i2.266

Günay, M. E., Türker, L., & Tapan, N. A. (2018). Decision tree analysis for efficient CO_2 utilization in electrochemical systems. *Journal of CO_2 Utilization, 28*, 83–95. https://doi.org/10.1016/j.jcou.2018.09.011

Frisch, M. J., Trucks, G. W., Schlegel, H. B., Scuseria, G. E., Robb, M. A., Cheeseman, J. R., Scalmani, G., Barone, V., Petersson, G. A., Nakatsuji, H., Li, X., Caricato, M., Marenich, A. V., Bloino, J., Janesko, B. G., Gomperts, R., Mennucci, B., Hratchian, H. P., Ortiz, J. V., Izmaylov, A. F., Sonnenberg, J. L., Williams-Young, D., Ding, F., Lipparini, F., Egidi, F., Goings, J., Peng, B., Petrone, A., Henderson, T., Ranasinghe, D., Zakrzewski, V. G., Gao, J., Rega, N., Zheng, G., Liang, W., Hada, M., Ehara, M., Toyota, K., Fukuda, R., Hasegawa, J., Ishida, M., Nakajima, T., Honda, Y., Kitao, O., Nakai, H., Vreven, T., Throssell, K., Montgomery Jr., J. A., Peralta, J. E., Ogliaro, F., Bearpark, M. J., Heyd, J. J., Brothers, E. N., Kudin, K. N., Staroverov, V. N., Keith, T. A., Kobayashi, R., Normand, J., Raghavachari, K., Rendell, A. P., Burant, J. C., Iyengar, S. S., Tomasi, J., Cossi, M., Millam, J. M., Klene, M., Adamo, C., Cammi, R., Ochterski, J. W., Martin, R. L., Morokuma, K., Farkas, O., Foresman, J. B., & Fox, D. J. (2016). *Gaussian 16* (Revision C.01) [Computer software]. Gaussian, Inc.

Martínez González, J. L. (2001). Optimizacion y ajuste de parametros mediante el metodo Simplex (Nelder–Mead). *EcosimPro User Workshop*, First EcosimPro User Workshop, 12.

Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: Modern heuristics* (2nd rev. ext. ed.). Springer.

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Mohammad, Y., Nakadai, S., & Greenwald, A. (2021). NegMAS: A platform for situated negotiations. In R. Aydoğan, T. Ito, A. Moustafa, T. Otsuka, & M. Zhang (Eds.), *Recent advances in agent-based negotiation* (Vol. 958, pp. 57–75). Springer Singapore. https://doi.org/10.1007/978-981-16-0471-3_4

Molina, M. M., Luna, J. M., Romero, C., & Ventura, S. (2012). Meta-learning approach for automatic parameter tuning: A case study with educational datasets. In *Proceedings of the Fifth International Conference on Educational Data Mining* (pp. 180–183).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). *Scikit-learn:*

Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. Retrieved from http://www.jmlr.org/papers/v12/pedregosa11a.html

Rivera Zárate, G. (2009). *Ajuste adaptativo de un algoritmo de enrutamiento de consultas semánticas en redes P2P* (Tesis de Maestría). Instituto Tecnológico de Cd. Madero.

Rodriguez-Fdez, I., Canosa, A., Mucientes, M., & Bugarín, A. (2015). STAC: A web platform for the comparison of algorithms using statistical tests. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ–IEEE)* (pp. 1–8). IEEE. https://doi.org/10.1109/FUZZ-IEEE.2015.7337889

Scalfani, V. F., Williams, A. J., Tkachenko, V., Karapetyan, K., Pshenichnov, A., Hanson, R. M., Liddie, J. M., & Bara, J. E. (2016). Programmatic conversion of crystal structures into 3D printable files using Jmol. *Journal of Cheminformatics, 8*, Article 66. https://doi.org/10.1186/s13321-016-0181-z

Srivastava, B., & Mediratta, A. (2005). Domain-dependent parameter selection of search-based algorithms compatible with user performance criteria. En *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)* (pp. 1386–1391). AAAI Press.

Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Prentice Hall.

Fukui, T., & Ito, T. (2018). A proposal of automatic negotiation simulator Jupiter for negotiation agent using machine learning. *Proceedings of the Annual Conference of the Japanese Society for Artificial Intelligence (JSAI2018)*. https://doi.org/10.11517/pjsai.JSAI2018.0_3Pin124

Villarreal-Hernández, J. Á., Morales-Rodríguez, M. L., Rangel-Valdez, N., & Gómez-Santillán, C. (2023). Reusability analysis of k-nearest neighbors variants for classification models. In G. Rivera, A. Rosete, B. Dorronsoro, & N. Rangel-Valdez (Eds.), *Innovations in machine and deep learning: Case studies and applications* (pp. 63–81). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-40688-1_4

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling, 28*(1), 31–36. https://doi.org/10.1021/ci00057a005

Young, D. C. (2002). *Computational chemistry: A practical guide for applying techniques to real world problems*. Wiley. http://onlinelibrary.wiley.com/book/10.1002/0471220655