Information Sciences 661 (2024) 120184

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

MOODY: An ontology-driven framework for standardizing multi-objective evolutionary algorithms

José F. Aldana-Martín *, María del Mar Roldán-García, Antonio J. Nebro, José F. Aldana-Montes

Dept. de Lenguajes y Ciencias de la Computación, ITIS Software, University of Málaga, ETSI Informática, Campus de Teatinos, Málaga, 29071, Spain

ARTICLE INFO

Keywords:
Ontology
Semantic technologies
Reasoning
Multi-objective optimization
Evolutionary algorithms

ABSTRACT

The application of semantic technologies, particularly ontologies, in the realm of multi-objective evolutionary algorithms is overlook despite their effectiveness in knowledge representation. In this paper, we introduce MOODY, an ontology specifically tailored to formalize these kinds of algorithms, encompassing their respective parameters, and multi-objective optimization problems based on a characterization of their search space landscapes. MOODY is designed to be particularly applicable in automatic algorithm configuration, which involves the search of the parameters of an optimization algorithm to optimize its performance. In this context, we observe a notable absence of standardized components, parameters, and related considerations, such as problem characteristics and algorithm configurations. This lack of standardization introduces difficulties in the selection of valid component combinations and in the re-use of algorithmic configurations between different algorithm implementations. MOODY offers a means to infuse semantic annotations into the configurations found by automatic tools, enabling efficient querying of the results and seamless integration across diverse sources through their incorporation into a knowledge graph. We validate our proposal by presenting four case studies.

1. Introduction

Multi-objective optimization problems are characterized by having two or more functions or objectives to be optimized (minimized or maximized) simultaneously. The objectives may conflict with each other, which means that the improvement of one of them leads to a worsening of the others [1]. As a consequence, the optimum of these kinds of problems is not usually a single solution but a set of trade-off solutions among the objectives known as the Pareto set, and the correspondence of this set in the objective space is the so-called Pareto front. Finding the Pareto set of a multi-objective problem is frequently unfeasible because, besides having more than one function, they can have properties such as non-linearity, NP-hard complexity, epistasis, etc. [2]. Consequently, the use of non-exact techniques has proliferated; in particular, metaheuristics [3] have become very popular. These constitute a broad family of optimization algorithms that do not guarantee to search for optimal solutions but can get quasi-optimal solutions in a reasonable time.

* Corresponding author.
E-mail addresses: jfaldanam@uma.es (J.F. Aldana-Martín), mmar@lcc.uma.es (M.d.M. Roldán-García), ajnebro@uma.es (A.J. Nebro), jfam@lcc.uma.es (J.F. Aldana-Montes).

https://doi.org/10.1016/j.ins.2024.120184
Received 6 June 2023; Received in revised form 19 December 2023; Accepted 18 January 2024

Available online 23 January 2024

When applied to solve multi-objective problems, metaheuristics are aimed then at finding an accurate approximation to the Pareto front. The accuracy is defined in terms of convergence (how close are the solutions to the Pareto front) and diversity (the solutions should be uniformly spread to cover all the region of the Pareto front); these properties are measured by quality indicators, such as the additive epsilon or the hypervolume [4]. Among metaheuristics, evolutionary algorithms are widely known and used techniques [5,6]. Popular multi-objective evolutionary algorithms include NSGA-II [7], SPEA2 [8], SMS-EMOA [9] or MOEA/D [10].

The search capabilities of metaheuristics are highly dependent on the particular settings of their control parameters. These parameters can be grouped into higher level components that determine the algorithm's behavior (e.g., the variation component defines how a metaheuristic introduces changes to the population during the algorithm execution), into algorithm hyperparameters (for example, the offspring population size in an evolutionary algorithm), or into component parameters (such as the probability of a crossover operator).

The main motivation driving us in this paper originated from our interest in applying auto-configuration [11] to evolutionary algorithms for multi-objective optimization. For this scenario, we have found several issues. First, there is no standard set of components to design these algorithms from, so each researcher working on the topic usually implements their own set of components, often similar to others previously defined. The lack of such a standard makes it difficult to compare data from different studies, as there is no framework to facilitate the integration of these heterogeneous data. Second, applying auto-configuration can be an intensive computing task (i.e., thousands of configurations can be generated and evaluated), so, given a particular problem, it would be interesting to find whether there is an existing configuration for a similar problem as this would save a significant amount of time. However, it is unclear how to specify problem properties and bind problems to configurations.

In this context, the use of semantic technologies has been proved in many other fields to integrate, represent domain knowledge, support data standardization and semantic integration from multiple sources [12]. Ontologies are the most widely spread method to describe the knowledge, especially allowing a formal and logic-based definition of concepts as a common vocabulary to share information in a specific domain [13].

Based on this necessity, this paper proposes a semantic framework to consolidate multi-objective optimization knowledge in the form of an OWL (Web Ontology Language) ontology following the FAIR principles (Findable, Accessible, Interoperable, and Reusable) [14]. Additionally, the proposed framework allows the exploitation of semantic reasoning [15] on the analysis of algorithm configurations or optimization experiments.

The main contributions of this paper are as follow:

- An ontology named MOODY (Multi-Objective Optimization ontologY)[1] for the formalization of multi-objective evolutionary algorithms, the parameters of said algorithms, multi-objective continuous problems with characteristics from the landscape of their search spaces and the quality indicators required to quantify the performance of the algorithms. The ontology formalizes the set of algorithm parameters for well-known evolutionary algorithms like NSGA-II or MOEA/D, which helps to integrate configurations of different algorithm implementations into a knowledge graph, and export them, if they are compatible, to a different implementation of the algorithm.
- We present a knowledge graph populated with algorithm configurations and optimization experiments, accompanied by semantic annotations adhering to the specified ontology and represented in RDF (Resource Description Framework) format.[2] This semantic framework empowers advanced reasoning capabilities over the knowledge graph, showcasing its efficacy in validating configurations of multi-objective algorithms within the domain of algorithm auto-configuration. Moreover, the knowledge graph serves as a foundation to provide users recommendations [16] of superior configurations of algorithms beyond their default settings.
- To validate the semantic approach, four use cases have been developed in the context of automatic configuration of multi-objective evolutionary algorithms: an in-depth description with examples on how MOODY can enrich an auto-configuration tool, the integration of algorithm configuration and experiments from different sources to validate new experiments, SPARQL (SPARQL Protocol And RDF Query Language) queries to obtaining valuable insights from the knowledge graph and integrating and exporting this knowledge into optimization frameworks used in real world applications, like pagmo [17] from the European Space Agency.

The rest of this work is structured as follows. In Section 2, key background concepts are described and a literature overview is given. Section 3 gives an in-depth description of the semantic approach followed, focusing on the ontology model. Four use cases to validate this approach are defined in Section 4, and are later discussed in Section 5. Finally, Section 6 provide the conclusions and future works.

2. Background and related work

This section describes background concepts about multi-objective evolutionary algorithms and the semantic technologies related to this work. A review of papers related to this work in the specialized literature is provided to add context to the contributions of our proposal concerning the current state of the art.

[1] Available on permanent URL: https://w3id.org/moody.

[2] The knowledge graph can be accessed at https://doi.org/10.5281/zenodo.7458095.

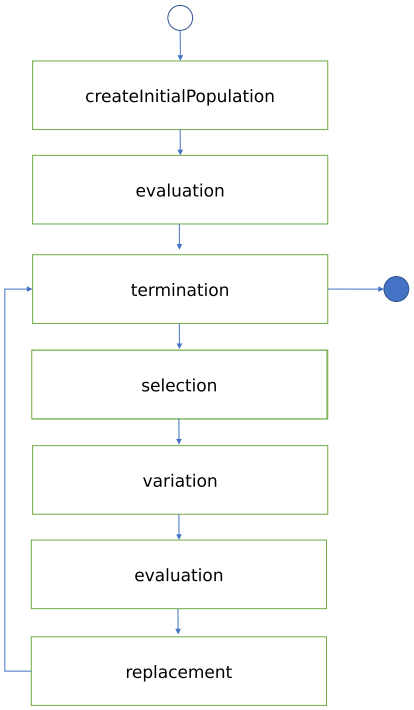

Fig. 1. Workflow representing an evolutionary algorithm.

2.1. Background concepts

```
P(0) ← GenerateInitialPopulation()
t ← 0
Evaluation(P(0))
while not TerminationCriterionIsMet() do
  P'(t) ← Selection(P(t))
  Q(t) ← Variation(P'(t))
  Evaluate(Q(t))
  P(t+1) ← Replacement(P(t), Q(t))
  t ← t+1
end while
```

Listing 1: Pseudo-code of an evolutionary algorithm.

- *Multi-Objective Evolutionary Algorithms.*

Evolutionary algorithms are widely used metaheuristics for dealing with multi-objective optimization problems [5]. They follow a generic behavior that follow the pseudo-code included in Listing 1. After creating an initial population of individuals, they are evaluated before starting the main loop of the algorithm, in which the typical steps of selection, variation (crossover and mutation) and replacement take place until a stopping condition is fulfilled.

The steps of a generic evolutionary algorithm can be designed as workflow of components, as depicted in Fig. 1. This way, a particular algorithm can be obtained by selecting particular individual components of each type. In the case of multi-objective evolutionary algorithms, replacement components typically include ranking strategies and density estimators, and the choice of using external archives (i.e., external populations) can be easily adopted by defining an evaluation component that stores in the archive any evaluated solution.

- *Ontology.* An ontology is a simplified and formal representation of a certain domain so that it can be represented for a purpose in terms of concepts and relationships [18]. The Web Ontology Language (OWL)[3] is a semantic markup language used to define and publish ontologies. OWL defines a set of representational primitives which are used to model a body of knowledge. These primitives are classes (or concepts), properties (or attributes), instances (or class members) and relationships. OWL is built on top of RDF and is a standard by the W3C. The syntax OWL-DL (Web Ontology Language - Description Logic) based on Description logic [19] has been used to formalize MOODY. Table 1 shows a summary of the OWL-DL syntax compared with the Manchester syntax [20].

[3] http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Table 1
Comparison between the OWL-DL semantic syntax used to formally define the proposed ontology compared with the Manchester syntax.

Descriptions	Abstract Syntax	DL Syntax	Manchester Syntax
Operators	$intersection(C_1, C_2, \cdots, C_n)$	$C_1 \sqcap C_2 \sqcap \cdots C_n$	C_1 and C_2 and $\cdots C_n$
	$union(C_1, C_2, \cdots, C_n)$	$C_1 \sqcup C_2 \sqcup \cdots C_n$	C_1 or C_2 or $\cdots C_n$
Restrictions	for at least 1 value V from C	$\exists V.C$	V some C
	for all values V from C	$\forall V.C$	V only C
	R is Symmetric	$R \equiv R^-$	R Characteristics: Symmetric
Class Axioms	$A\ partial(C_1, C_2, \cdots, C_n)$	$A \sqsubseteq C_1 \sqcap C_2 \sqcap \cdots C_n$	A EquivalentTo: C_1 and C_2 and $\cdots C_n$
	$A\ complete(C_1, C_2, \cdots, C_n)$	$A \equiv C_1 \sqcap C_2 \sqcap \cdots C_n$	A SubClassOf: C_1 and C_2 and $\cdots C_n$

- *RDF.* Resource Description Framework is a graph-based markup language used to represent information in the Web,[4] being defined as a W3C standard. W3C encourages the use of RDF in applications where data is shared and processed by other machines and not only end-users. RDF uses URIs (Uniform Resource Identifier) to identify each resource in the Web. Statements in the form of triples, which contain a subject, a predicate and an object [13], are used to define each resource. RDF Schema (RDFS) extends RDF to allow the use of classes and properties of resources [13].

- *SPARQL.* SPARQL is a query language for graphs,[5] allowing the extraction and manipulation of information in RDF format over web resources identified via URIs. SPARQL queries use graph-matching to retrieve the set of RDF triples that match a pattern.

- *SWRL.* The Semantic Web Rule Language provides ontologies defined in OWL additional ways to describe semantic relationships between individuals adding extra inference capabilities [21]. SWRL is based on rule expressions in the form of "Antecedent ⇒ Consequent" to represent semantic relationships. The antecedent and the consequent can be formulated as conjunctions of elements associated with one or more attributes defined by a question mark and a variable (e.g., $?x$) in the rule.

2.2. Related works

The primary objective of the proposed ontology is to formalize algorithms, parameters, problems, and quality indicators within the multi-objective optimization domain. This formalization aims to facilitate the creation of a knowledge graph comprising algorithm configurations, capable of assimilating experiments from diverse and heterogeneous sources, empowering semantic reasoning over this knowledge graph.

The use of ontologies to formalize knowledge has not yet been widely accepted in the field of multi-objective optimization. To the best of our knowledge, the only ontology on multi-objective optimization is PMOEA ontology (Preference-based Multi-Objective Evolutionary Algorithm) [22,23]. PMOEA focuses mainly on preference-based algorithms, preference models and preference integration to guide the algorithms. This focus on a small subset of the evolutionary algorithms field makes this ontology not broad enough for our proposed framework.

There are other published ontologies with a more focused scope inside multi-objective optimization. Examples of these ontologies are for modeling constraints in multi-objective optimization algorithms [24] or the domain knowledge of the field of the multi-objective problem [25–27]. These ontologies are transversal to MOODY. MOODY focuses on the formalization of the algorithms themselves, while the ontologies mentioned here attempt to use domain knowledge to help solve a problem in a specific domain.

One ontology with similar objectives of MOODY is OPTION (OPTImization algorithm benchmarking ONtology) [28], which is aimed at making benchmarking more reusable and interoperable, but it is centered only on single-objective optimization. OPTION and MOODY are complementary, each focusing on a different part of the optimization field, so we have established mappings between them (see next section).

3. Semantic approach

MOODY is implemented as an OWL 2 ontology following the FAIR principles and designed following the *Ontology Development 101* methodology [29] as follows:

1. **Determine the domain and scope of the ontology**. The goal of MOODY is to be able to model experimentation in multi-objective optimization problems and algorithms. It provides a framework that allows algorithms to be formalized with all their configurable parameters and problems defined by their defining characteristics. The scope of MOODY is focused on the most relevant algorithms (NSGA-II [7], MOEA/D [10] and MOEA/D-DE [30]) and benchmark problems (Deb-Thiele-Laumanns-Zitzler (DTLZ) [31], Gu-Liu-Tan (GLT) [32], Li-Zhang 2009 (LZ09) [30], REal world problems (RE) [33], CEC 2009 competition

[4] https://www.w3.org/TR/rdf11-primer/.
[5] https://www.w3.org/TR/sparql11-query/.

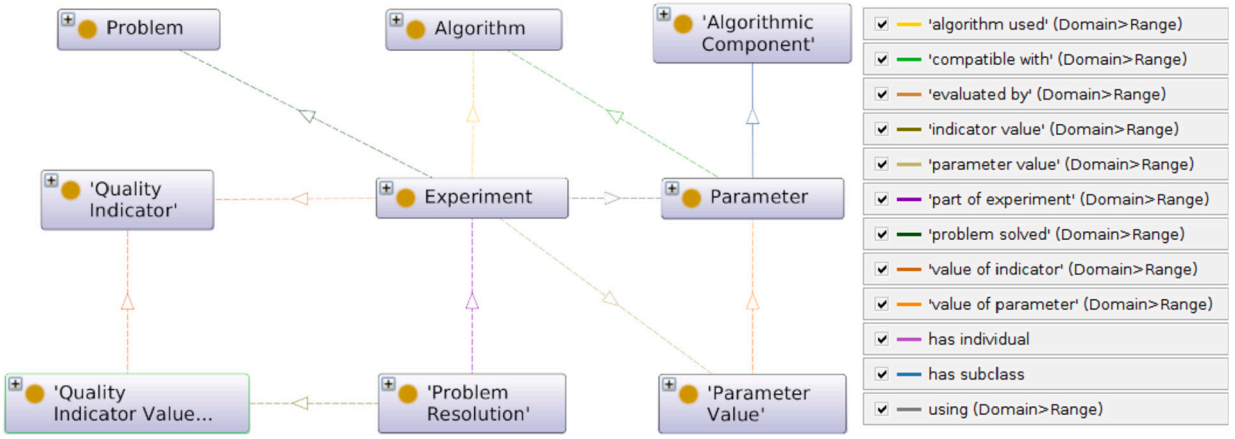

Fig. 2. Class diagram of MOODY. Continuous arrows refer to subclass of. Dotted arrows refer to specific properties as shown in the figure legend at right.

(UF) [34], Walking-Fish-Group (WFG) [35] and Zitzler-Deb-Thiele (ZDT) [36] problem families) to highlight the features of the ontology, but MOODY is designed to be easily extended with new algorithms and real world problems.

2. **Consider reusing existing ontologies**. To align our proposal with existing vocabularies, several ontologies have been reused. Firstly, the Data Mining Optimization Ontology (DMOP) [37] has been used to describe the parameters for the optimization algorithms. From the big data analytics OWL2 ontology (BIGOWL) [38], the definition of the concepts of algorithm and problem is reused. The OPTImization algorithm benchmarking ONtology (OPTION) [28] is modeling single-objective optimization, but some terms are common on both domains, so mappings have been created between them, such as *moody:QualityIndicator* and *ontoopt:performance_evaluation_function* or *moody:ProblemResolution* and *ontoopt:algorithm_execution*.
3. **Enumerate important terms in the ontology**. The main terms of MOODY are the *Experiment*, the *Algorithm*, the *Parameter*, the *Problem*, the *Quality Indicator* and the *Problem Resolution*, which includes the results of each execution in the *Experiment*.
4. **Define classes and the class hierarchy**. The classes on the ontology model are the key concepts defined in the previous point and more specific classes to model concrete *Algorithms* or *Problems*. Fig. 2 shows all the classes defining the key concepts. For example, *Algorithm* is a concept defined at BIGOWL, but MOODY expands it with the subclasses *NSGA-II* and *MOEA/D*. These high level classes allow the ontological model to be easily expanded when a use case needs new, more specific algorithms. They can be easily integrated as new subclasses of *Algorithm*, inheriting all the semantic relationships from it.
5. **Define the properties of classes and slots**. Object properties are used to define relationships between classes. For example, an *Experiment* is *evaluated by* a *Quality Indicator* or a *Parameter* is *compatible with* an *Algorithm*. Data properties are used to define what attributes an instance of a class has. One example of data properties is *Algorithm* which has a *crossover* type, a *population size*, between others.
6. **Define the facets of the slots**. All properties in the ontology are constrained by their range and domain; for example, the object property *evaluated by* has a range of *Experiment* and a domain of *Quality Indicator*. For the data properties, the domain is specified at the parent property, while the range is specific to each property. Data properties range is not defined only by its type but also by the possible values. These range definitions allow a semantic reasoner, like Pellet [39], to validate if an algorithm configuration is valid, as showcased in Section 4. Table 2 shows the range of all algorithm parameters.
7. **Create instances**. Individuals are specific data points that belong to a class. These instances are created by mapping experimentation data to RDF by following the model defined by the ontology. In our case, the auto-configuration tool irace [40] has been used to generate instances of configurations and map them to RDF creating a knowledge graph.

3.1. Ontology model

The MOODY ontology has a total of 56 classes, 11 object properties, 78 data properties and 561 logical axioms. The focus of the proposed ontology is bifold; on the one hand, the ontology formally defines multi-objective optimization problems, the algorithms used to solve them and the quality indicators used to validate the quality of the solutions. On the other hand, MOODY provides a framework for the semantic annotation of multi-objective optimization experiments.

Fig. 2 depicts the main classes of MOODY and Table 3 shows the formal definition in description logic of the object properties in the ontology. A short description of the most important classes is included next:

- ***Problem*** of multi-objective optimization, meant to be solved in a *Experiment*. MOODY supports characterizing each problem through a series landscape characteristics [41] obtained through a sampling of the search space. Some of the characteristics included are: the number of variables and objectives, the average and maximum distance among solutions in the variable and objective space, the proportion of non-dominated solutions or the average proportion of dominated solutions between neighbors. MOODY currently contains the following families of problems with their characteristics: DTLZ, GLT, LZ09, RE, UF, WFG and ZDT.

Table 2
Ranges for the possible parameters of an algorithm. The domain of all parameters is *Parameter Value*. Type of a value is xsd:string unless specified.

Parameter	range
Aggregative function	{“PenaltyBoundaryIntersection”, “Tschebycheff”, “WeightedSum”}
Algorithm result	{“externalArchive”, “population”}
BLX alpha crossover alpha value	xsd:double[>= “0.0”^^xsd:double, <= “1.0”^^xsd:double]
Create initial solutions	{“latinHypercubeSampling”, “random”, “scatterSearch”}
Crossover probability	xsd:double[>= “0.0”^^xsd:double, <= “1.0”^^xsd:double]
Crossover repair strategy	{“bounds”, “random”, “round”}
Crossover	{“BLX_ALPHA”, “SBX”}
Maximum number of evaluations	xsd:integer
Maximum number of replaced solutions	xsd:integer
Mutation probability	xsd:double[>= “0.0”^^xsd:double, <= “1.0”^^xsd:double]
Mutation repair strategy	{“bounds”, “random”, “round”}
Mutation	{“polynomial”, “uniform”}
Neighborhood selection probability	xsd:double[>= “0.0”^^xsd:double, <= “1.0”^^xsd:double]
Neighborhood size	xsd:integer
Offspring population size	xsd:integer
Polynomial mutation distribution index	xsd:double[>= “5.0”^^xsd:double, <= “400.0”^^xsd:double]
Population size	xsd:integer
Population size with archive	xsd:integer
SBX crossover distribution index	xsd:double[>= “5.0”^^xsd:double, <= “400.0”^^xsd:double]
Selection tournament size	xsd:integer[>=2, <=10]
Selection	{“random”, “tournament”}
Uniform mutation perturbation	xsd:double[>= “0.0”^^xsd:double, <= “1.0”^^xsd:double]

Table 3
List of the most relevant object properties in MOODY formally defined using the description logic syntax.

Object Properties	Description Logic
algorithmUsed	$\exists$ algorithmUsed Thing $\sqsubseteq$ Experiment
	$\top \sqsubseteq \forall$ algorithmUsed Algorithm
compatibleWith	$\exists$ compatibleWith Thing $\sqsubseteq$ Parameter
	$\top \sqsubseteq \forall$ compatibleWith Algorithm
evaluatedBy	$\exists$ evaluatedBy Thing $\sqsubseteq$ Experiment
	$\top \sqsubseteq \forall$ evaluatedBy QualityIndicator
indicatorValue	$\exists$ indicatorValue Thing $\sqsubseteq$ ProblemResolution
	$\top \sqsubseteq \forall$ indicatorValue QualityIndicatorValue
parameterValue	$\exists$ parameterValue Thing $\sqsubseteq$ Experiment
	$\top \sqsubseteq \forall$ parameterValue ParameterValue
partOfExperiment	$\exists$ partOfExperiment Thing $\sqsubseteq$ ProblemResolution
	$\top \sqsubseteq \forall$ partOfExperiment Experiment
problemSolved	$\exists$ problemSolved Thing $\sqsubseteq$ Experiment
	$\top \sqsubseteq \forall$ problemSolved Problem
using	$\exists$ using Thing $\sqsubseteq$ Experiment
	$\top \sqsubseteq \forall$ using Parameter
valueOfIndicator	$\exists$ valueOfIndicator Thing $\sqsubseteq$ QualityIndicatorValue
	$\top \sqsubseteq \forall$ valueOfIndicator QualityIndicator
valueOfParameter	$\exists$ valueOfParameter Thing $\sqsubseteq$ ParameterValue
	$\top \sqsubseteq \forall$ valueOfParameter Parameter

- *Algorithm* to solve a multi-objective optimization *Problem*, and it is used in a *Experiment*. Each *Algorithm* is compatible with a series of *Parameters* that modify its behavior. Widespread algorithms from the state of the art have been added to the ontology as a reference, like NSGA-II, MOEA/D and MOEA/D-DE.

- *Parameter* of an *Algorithm*. Implementations of metaheuristics allow changing the components of the *Algorithm*, such as changing the crossover function, without developing an entirely new *Algorithm*. This flexibility in configurations is particularly useful in the application of techniques of auto-configuration of algorithms to optimize a metaheuristic to a specific problem (or set of problems). In the current literature, each implementation defines its parameters, making it difficult to compare two implementations. MOODY aims to provide a standard definition of the parameters of the algorithms from the state of the art. *Parameters* are compatible with a series of *Algorithms* and are used in a *Experiment*.

- *Algorithmic Component* is a subclass of *Parameter* that model the main components of a metaheuristic. This *Algorithmic Components* are: the creation of initial solutions, termination criterion, selection, variation and replacement components. How this components form on a generic evolutionary algorithm is described at Listing 1.

- *Quality Indicators* are used as metrics to quantify the quality of the non-dominated front after the process of optimization by a metaheuristic. They evaluate each *Problem Resolution* in a *Experiment*. The indicators usually measure the current front’s quality

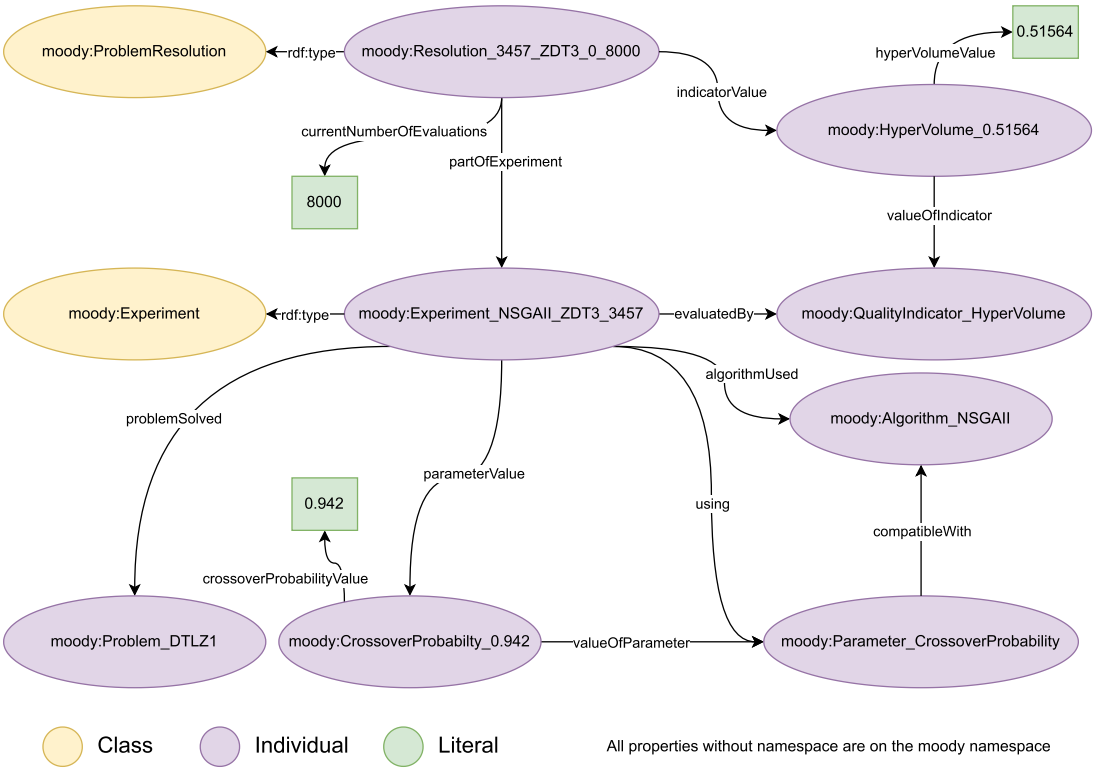

Fig. 3. Graph representation of a knowledge graph.

in terms of convergence, diversity or both. Currently, the most common quality indicators for MOPs have been included: Hypervolume [42], Spread [7], Inverted Generational Distance [43], Inverted Generational Distance Plus [44] and Epsilon [45].

- ***Problem Resolution*** defines a single execution of an *Experiment*. It is the execution of a single *Algorithm*, using a defined set of *Parameters*, to solve a specific *Problem*. This execution is evaluated with one or more *Quality Indicator*, having one single *Quality Indicator Value* each.

- ***Experiment*** defines a series of independent *Problem Resolution*, utilizing one *Algorithm* with a fix set of *Parameters* to solve a *Problem* while evaluated by a series of *Quality Indicators*.

3.2. Data consolidation

Once the ontology is defined, it is linked with other ontologies like DMOP, BIGOWL, and OPTION. The linked ontology constitutes the terminological box (TBox) of the proposed semantic framework. The assertional box (ABox) considers all the instances containing the data modeled. The ABox is built as a knowledge graph guided by the ontology. In order to create a solid base for the knowledge graph, we applied the auto-configuration framework irace with jMetal [46] to produce configurations for the problems mentioned in the previous section, standardized to RDF format using mapping functions.

The knowledge graph is then stored in an instance of Stardog[6] repository for persistence and reasoning capabilities. A subset of the knowledge graph is shown in Fig. 3 as an example of a configuration graph structure. Fig. 4 provides an overview of the semantic model[7] and how data gets ingested into the knowledge graph. In Section 4.2, we illustrate a sample case demonstrating the process of transforming an algorithmic configuration into a knowledge graph according to the MOODY ontology knowledge.

A reference implementation of a tool to ingest into RDF is provided.[8] This reference implementation in Python demonstrates how to load to the knowledge graph configurations generated from irace 3.4.1 and the jMetal framework. The Python script processes each configuration from the irace output. It incorporates each configuration into the knowledge graph as an Experiment, specifying their properties in accordance with the structure defined by the ontology. This ensures that each configuration's details are captured and formally integrated into the broader context of the knowledge graph.

[6] https://www.stardog.com.

[7] The linked Open Data Cloud comes from https://lod-cloud.net/.

[8] Reference implementation of mapping function available at https://doi.org/10.5281/zenodo.7458095.

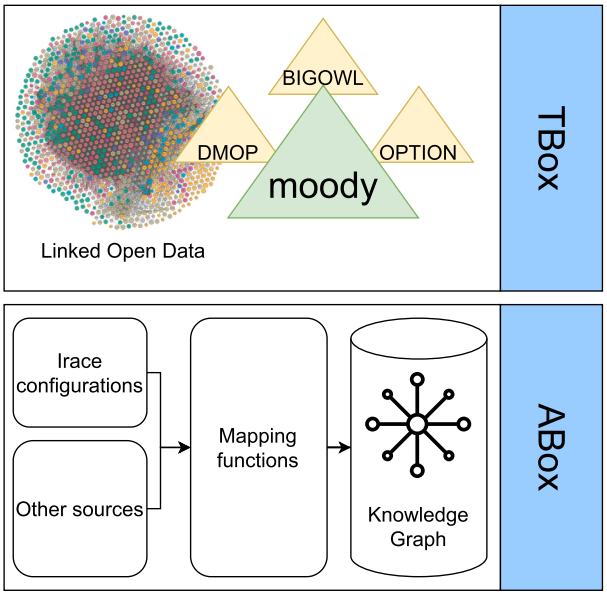

Fig. 4. General overview of the semantic framework MOODY.

4. Validation

Four defined use cases validate the proposed MOODY ontology, highlighting its key features. These use cases focus tasks such as performing semantic validation of algorithm configurations, integrating existing knowledge from prior studies for new experiments, executing sample queries on the resultant knowledge graph to obtain valuable insights from the data, and, finally, exporting configurations to various implementations.

4.1. Supporting auto-configuration

During the auto-configuration process, tools typically generate hundreds and thousands of configurations, and some of them may lack significant differences. This can occur due to various reasons, including negligible distinctions between continuous parameters or, depending on the tool, a lack of consideration for dependencies among parameters. For instance, it may evaluate two configurations that utilize random selection but differ in the selection tournament size, even though the tournament size parameter is only relevant when the tournament component is employed as the selection method.

In this context, the utilization of a semantic model and the creation of a knowledge graph can provide interesting options for auto-configuration packages. Firstly, by leveraging the knowledge graph populated with past executions and the formal specification of parameter hierarchy, we can utilize previous knowledge to avoid executing configurations that are known to yield poor results. Furthermore, by utilizing knowledge of previous executions as a starting point, a semantically-enriched auto-configuration tool can enhance the initial state and facilitate more efficient configuration processes.

Secondly, having a formalized definition for the algorithm and its parameters allows a SWRL engine to execute semantic reasoning and infer new knowledge from the data we annotate semantically. There are two ways to use a semantic reasoner to validate configurations: one is to validate that every property has a valid domain and range and the other is to use a reasoner to validate algorithm configurations automatically. Due to the Open World Assumption, we cannot check directly if a configuration is valid. To validate a configuration, we need to create SWRL rules that assert whether a configuration is invalid.

For the validation of the domain and range of the properties, MOODY formalizes both data and object properties (examples of each are described in Tables 2 and 3, respectively). A semantic reasoner will check if those restrictions are met, and if they are not, it will point out where the inconsistency lies.

For the automatic validation of configurations, the SWRL rules defined in Listings 2 and 3 check cases where a configuration is invalid because it mixes configuration parameters of two crossover operators, SBX and BLX_Alpha. Listing 2 reads as follows: “Being **?x** an experiment with a parameter **?p2** of type Crossover and a parameter **?p** of type SbxCrossoverDistributionIndex if both parameters have a value, and the value of Crossover is ‘BLX_ALPHA’, then **?x** is an invalid configuration”. This rule checks that parameters that depend on the SBX crossover are not used with the BLX_Alpha crossover, as they are incompatible.

Integrating these two approaches into auto-configuration tools enables the acceleration of the execution process by pruning sections of the search space that have previously been explored with unfavorable outcomes or being discarded due to semantic reasoning. Moreover, it guarantees the validity of the configurations being evaluated.

```
moody:Experiment(?x)
^ moody:SbxCrossoverDistributionIndex(?p)
^ moody:sbxCrossoverDistributionIndexValue(?pv, ?cv)
^ moody:parameterValue(?x, ?pv)
^ moody:valueOfParameter(?pv, ?p)
^ moody:Crossover(?p2)
^ moody:crossoverValue(?pv2, ?cv2)
^ moody:parameterValue(?x, ?pv2)
^ moody:valueOfParameter(?pv2, ?p2)
^ swrlb:equal(?cv2, "BLX_ALPHA")
-> moody:InvalidExperiment(?x)
```

Listing 2: A SWRL rule to validate when parameters that depend from the SBX crossover are used on an algorithm that uses BLX_Alpha as the crossover operation.

```
moody:Experiment(?x)
^ moody:BlxAlphaCrossoverAlphaValue(?p)
^ moody:blxAlphaCrossoverAlphaValueValue(?pv, ?cv)
^ moody:parameterValue(?x, ?pv)
^ moody:valueOfParameter(?pv, ?p)
^ moody:Crossover(?p2)
^ moody:crossoverValue(?pv2, ?cv2)
^ moody:parameterValue(?x, ?pv2)
^ moody:valueOfParameter(?pv2, ?p2)
^ swrlb:equal(?cv2, "SBX")
-> moody:InvalidExperiment(?x)
```

Listing 3: A SWRL rule to validate when parameters that depend from the BLX_Alpha crossover are used on an algorithm that uses SBX as the crossover operation.

4.2. Data integration

Semantic technologies allow data integration from several sources in a standardized format, RDF. We can semantically annotate the configurations provided in a previous study using an ontology to model the data. We include the study presented in [47], where NSGA-II is auto-configured using the WFG family of problems and later used to solve the WFG and DTLZ families against the default configuration of NSGA-II and other metaheuristics. Listing 4 shows the configuration of the NSGA-II found in the study and the result of the algorithm solving the DTLZ1 problem with it in RDF serialized as turtle. Fig. 5 shows a subset of this configuration as a graph. The ingesting of this data to the knowledge graph depends on how the configurations were defined in the original source. However, custom mapping functions or R2RML[9] rules can be defined to facilitate the process.

As previously mentioned, the auto-configuration of algorithms yields a substantial volume of configurations. Employing mapping functions to assimilate these configurations into a knowledge graph, guided by the MOODY semantic framework, opens avenues for in-depth analysis, such as validation of the configurations, further reasoning analysis and the execution of SPARQL queries over them.

4.3. Querying the knowledge graph

Once the data for a series of experiments have been ingested into the knowledge graph guided by the ontology, we can use the semantic model to query the data using the SPARQL query language. This use case shows sample SPARQL queries that can be used to get insights into the knowledge graph. One first sample in Table 4 includes the SPARQL query defined to obtain the parameter configuration from a specific experiment. This query is a common yet useful query as most other analyses will only return the experiment URI.

When experimenting with new algorithm configurations to solve a particular problem, you can consult the knowledge base to find the best configurations that have been discovered. The second query, as described in Table 5, provides the answer to the question of which algorithmic configuration is most effective in solving a specific problem, based on a particular quality indicator. This query takes into account that metaheuristics, in general, are not deterministic, so it calculates the average value from multiple experiment executions.

A derived query to check for the best performing algorithm for problems with similar characteristics can be seen in Table 6 for problems with a disconnected front. Comparing problems by their landscape helps transfer the knowledge of good configurations from one problem to a new, similar problem. An open line of research is the optimal way to define groups of problems with a similar landscape.

[9] https://www.w3.org/TR/r2rml/.

```
###  https://w3id.org/moody#Experiment_NSGAII_DTLZ1_GECCO19
moody:Experiment_NSGAII_DTLZ1_GECCO19 rdf:type owl:NamedIndividual ,
        moody:Experiment ;
    moody:algorithmUsed moody:Algorithm_NSGAII ;
    moody:evaluatedBy moody:QualityIndicator_HyperVolume ;
    moody:parameterValue moody:ParameterValue_AlgorithmResult_externalArchive ,
        moody:ParameterValue_BlxAlphaCrossoverAlphaValue_0.5906 ,
        moody:ParameterValue_CreateInitialSolutions_latinHypercubeSampling ,
        moody:ParameterValue_CrossoverProbability_0.9874 ,
        moody:ParameterValue_CrossoverRepairStrategy_bounds ,
        moody:ParameterValue_Crossover_BLX_ALPHA ,
        moody:ParameterValue_ExternalArchive_crowdingDistanceArchive ,
        moody:ParameterValue_MaximumNumberOfEvaluations_25000 ,
        moody:ParameterValue_MutationProbability_0.0015 ,
        moody:ParameterValue_MutationRepairStrategy_random ,
        moody:ParameterValue_Mutation_polynomial ,
        moody:ParameterValue_OffspringPopulationSize_200 ,
        moody:ParameterValue_PolynomialMutationDistributionIndex_158.05 ,
        moody:ParameterValue_PopulationSizeWithArchive_20 ,
        moody:ParameterValue_PopulationSize_100 ,
        moody:ParameterValue_ProblemName_dtlz.DTLZ1 ,
        moody:ParameterValue_ReferenceFrontFileName_DTLZ1.csv ,
        moody:ParameterValue_SelectionTournamentSize_9 ,
        moody:ParameterValue_Selection_tournament ,
        moody:ParameterValue_Variation_crossoverAndMutationVariation ;
    moody:problemSolved moody:Problem_DTLZ1 ;
    moody:using moody:Parameter_AlgorithmResult ,
        moody:Parameter_BlxAlphaCrossoverAlphaValue ,
        moody:Parameter_CreateInitialSolutions ,
        moody:Parameter_Crossover ,
        moody:Parameter_CrossoverProbability ,
        moody:Parameter_CrossoverRepairStrategy ,
        moody:Parameter_ExternalArchive ,
        moody:Parameter_MaximumNumberOfEvaluations ,
        moody:Parameter_Mutation ,
        moody:Parameter_MutationProbability ,
        moody:Parameter_MutationRepairStrategy ,
        moody:Parameter_OffspringPopulationSize ,
        moody:Parameter_PolynomialMutationDistributionIndex ,
        moody:Parameter_PopulationSize ,
        moody:Parameter_PopulationSizeWithArchive ,
        moody:Parameter_ProblemName ,
        moody:Parameter_ReferenceFrontFileName ,
        moody:Parameter_Selection ,
        moody:Parameter_SelectionTournamentSize ,
        moody:Parameter_Variation .

###  https://w3id.org/moody#Resolution_GECCO19_DTLZ1_0_25000
moody:Resolution_GECCO19_DTLZ1_0_25000 rdf:type owl:NamedIndividual ,
        moody:ProblemResolution ;
    moody:indicatorValue moody:QualityIndicatorValue_HyperVolume_0.00353 ;
    moody:partOfExperiment moody:Experiment_NSGAII_DTLZ1_GECCO19 ;
    moody:currentNumberOfEvaluations 25000 .
```

Listing 4: RDF data of a NSGA-II configuration to solve the DTLZ1 problem in turtle format.

4.4. Exporting configurations to different frameworks

As we mentioned in Section 4.2, we can integrate configurations from diverse sources. However, the formal specification provided by MOODY allows to extract the best configuration that can be compatible with different implementations of an algorithm.

In this use case, we devise a scenario in which we have a knowledge graph populated with configurations of the NSGA-II multi-objective evolutionary algorithm obtained with different sources (such as irace and jMetal, as mentioned before) for a set optimization problems. In this context, a user is interested in using the NSGA-II algorithm included in pagmo, a framework for massively parallel optimization developed by the European Space Agency [17]; concretely, the user would like to get a configuration of NSGA-II for a particular problem (ZDT4). So, instead of try to find that configuration using pilot tests, that is a trial-and-error process, or using an automatic configuration tool (which can take hours of computation), an alternative is to query the knowledge graph for a stored configuration of NSGA-II for that problem.

The NSGA-II implementation of pagmo only allows to set the parameters of the standard NSGA-II for continuous optimization, which are restricted to the distribution indices of the SBX crossover and Polynomial mutation and the mutation and crossover probabilities. These parameters are a subset of the parameters we are annotating in the ontology, as such when querying for the result we extend the previous queries to find configurations that use the values of the standard version of NSGA-II. Table 7 demonstrate how we can query for configurations that are compatible with pagmo. Small tolerances are given to floating point values to ensure that all relevant configurations are being retrieved.

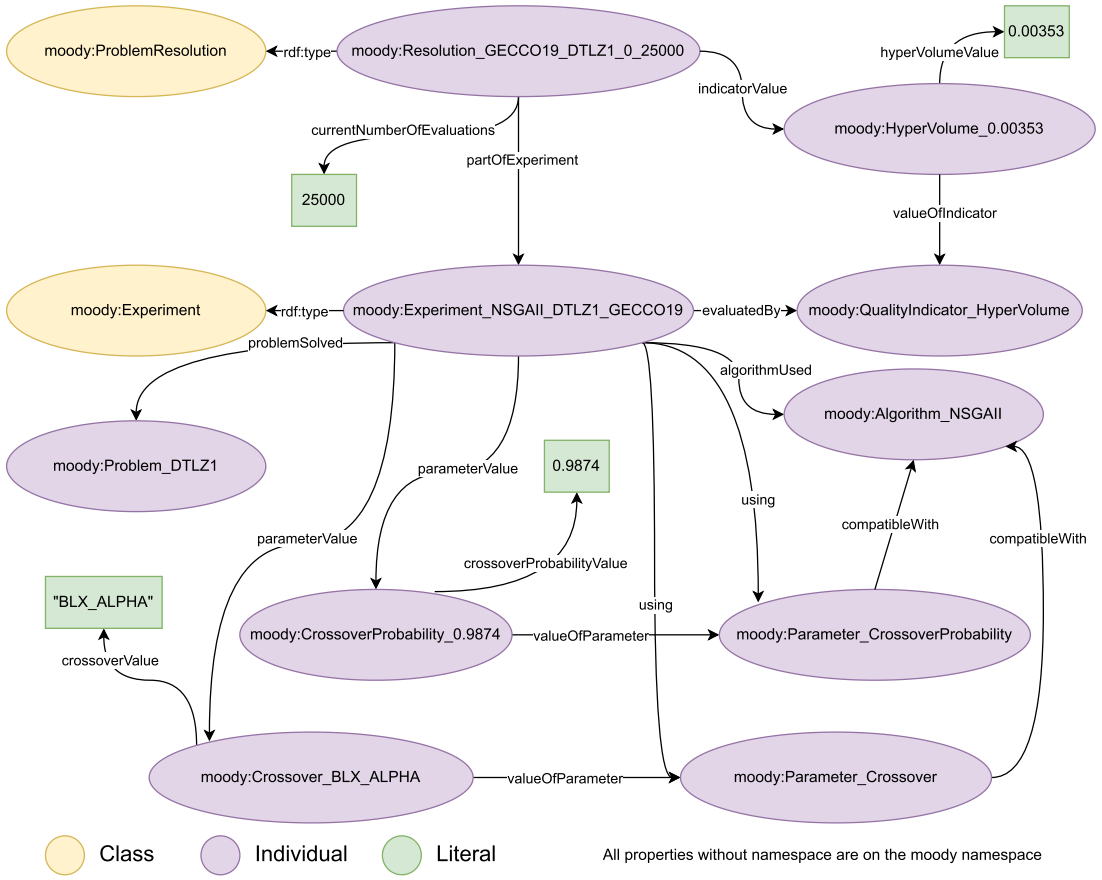

Fig. 5. Graph representation of a subset of a NSGA-II configuration to solve the DTLZ1 problem.

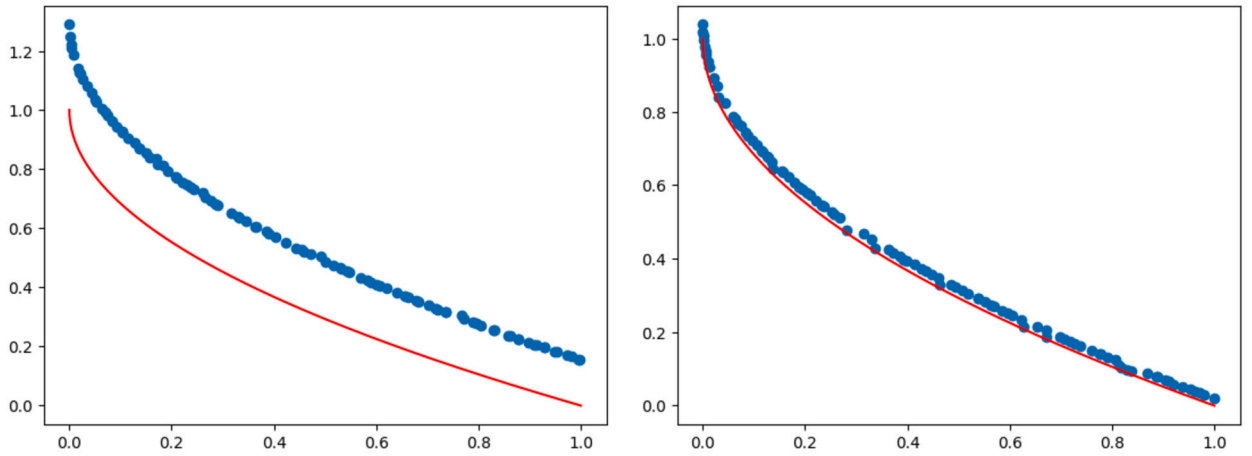

Fig. 6. In red, reference front for the ZDT problem. In blue, Front for the ZDT4 problem obtained by NSGA-II with standard settings (left), and front obtained by exporting a configuration from MOODY (right). The target framework is pagmo.

Fig. 6 shows an example of the fronts that can be obtained with the NSGA-II included in pagmo for the considered problem. The front on the left is obtained using the standard NSGA-II settings and the front on the right is the corresponding one to run the algorithm using the configuration returned by MOODY.

5. Discussion

There are two main reasons for the use of semantic technologies to model the configurations of evolutionary algorithms. First, they are open standards for semantic technologies defined by the W3C for the publication and integration of data and are supported by

Table 4
SPARQL Query Q1, extraction the parameters of one specific experiment. **moody:Experiment_NSGAII_DTLZ1_GECCO19** is an example of the URI of an *Experiment*.

```
PREFIX moody: <https://w3id.org/moody#>
SELECT DISTINCT ?parameter ?value
WHERE {
    VALUES ?experiment { moody:Experiment_NSGAII_DTLZ1_GECCO19 } .
    ?experiment moody:parameterValue ?parameter_value .
    ?parameter_value moody:valueOfParameter ?parameter .
    ?parameter_value ?property ?value .
    ?property rdfs:subPropertyOf moody:parameterValueProperty .
}
```

Result:

?parameter	?value
moody:Parameter_AlgorithmResult	externalArchive
moody:Parameter_BlxAlphaCrossoverAlphaValue	0.5906
moody:Parameter_CreateInitialSolutions	latinHypercubeSampling
moody:Parameter_CrossoverProbability	0.9874
moody:Parameter_CrossoverRepairStrategy	bounds
moody:Parameter_Crossover	BLX_ALPHA
moody:Parameter_ExternalArchive	crowdingDistanceArchive
moody:Parameter_MaximumNumberOfEvaluations	25000
moody:Parameter_MutationProbability	0.0015
moody:Parameter_MutationRepairStrategy	random
moody:Parameter_Mutation	polynomial
moody:Parameter_OffspringPopulationSize	200
moody:Parameter_PolynomialMutationDistributionIndex	158.05
moody:Parameter_PopulationSizeWithArchive	20
moody:Parameter_PopulationSize	100
moody:Parameter_ProblemName	dtlz.DTLZ1
moody:Parameter_ReferenceFrontFileName	DTLZ1.csv
moody:Parameter_SelectionTournamentSize	9
moody:Parameter_Selection	tournament
moody:Parameter_Variation	crossoverAndMutationVariation

Table 5
SPARQL Query Q2, recover the experiments that better resolve a problem according to a specific quality indicator. **moody:Problem_ZDT2** is the URI of a *Problem* and **moody:QualityIndicator_HyperVolume** is the URI of a *Quality Indicator*.

```
PREFIX moody: <https://w3id.org/moody#>
SELECT DISTINCT ?experiment (AVG(?exp_value) as ?value)
WHERE {
    VALUES ?problem { moody:Problem_ZDT2 } .
    VALUES ?indicator { moody:QualityIndicator_HyperVolume } .
    ?problem_resolution rdf:type moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?problem_resolution moody:indicatorValue ?indicatorValue .
    ?indicatorValue moody:valueOfIndicator ?indicator .
    ?indicatorValue moody:hyperVolumeValue ?exp_value .
}
GROUP BY (?experiment)
ORDER BY ASC(?value)
LIMIT 5
```

Result:

?experiment	?value
moody:Experiment_2246	3.28790E-1
moody:Experiment_2235	3.28792E-1
moody:Experiment_2241	3.28801E-1
moody:Experiment_1935	3.28809E-1
moody:Experiment_2266	3.28814E-1

open tools, allowing different optimization frameworks or implementations to store or retrieve configurations from a implementation-agnostic knowledge graph, as well as exporting them to each implementation via mapping functions. The other driving factor behind the use of semantic web technologies is to present data in a format that can be utilized not only by humans, but also by other tools or applications.

Table 6
SPARQL Query Q3, recover the experiments that better resolve any problem with a disconnected front.

```
PREFIX moody: <https://w3id.org/moody#>
PREFIX bigowl: <http://www.khaos.uma.es/lod/bigowl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?experiment (SAMPLE(?problem) as ?problem) (MIN(?exp_value) as ?value)
WHERE {
    VALUES ?geometry { "disconnected"^^xsd:string  } .
    VALUES ?indicator { moody:QualityIndicator_HyperVolume } .
    ?problem_resolution rdf:type moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?problem rdf:type bigowl:Problem .
    ?probem moody:geometry ?geometry .
    ?problem_resolution moody:indicatorValue ?indicatorValue .
    ?indicatorValue moody:valueOfIndicator ?indicator .
    ?indicatorValue moody:hyperVolumeValue ?exp_value .
}
GROUP BY (?experiment)
ORDER BY ASC(?value)
LIMIT 5
```

Result:

?experiment	?problem	?value
moody:Experiment_3049	moody:Problem_ZDT4	6.58993E-1
moody:Experiment_1731	moody:Problem_ZDT4	6.59747E-1
moody:Experiment_3338	moody:Problem_ZDT4	6.60115E-1
moody:Experiment_3541	moody:Problem_ZDT4	6.60986E-1
moody:Experiment_3171	moody:Problem_ZDT4	6.61028E-1

While these open standards are widely utilized, it is important to note that not all practitioners may be familiar with them. Consequently, we believe that the primary limitation of our semantic framework is the requirement for users to acquaint themselves with these standards in order to integrate their applications or tools with MOODY. However, from our point of view the benefits of said standards (automatic integration or open tools like reasoners) make it worthwhile for our framework.

The field of metaheuristics sees a continuous influx of algorithm proposals, often referred as new techniques, though in many cases they are variations of metaphor-based approaches like evolutionary algorithms, particle swarm optimization, or ant colony optimization [48–50]. In this sense, the use of an ontology such as MOODY can be the basis of a formalization of metaheuristic families, including their constituting components and their relationships. This would help in detecting when a new proposal is in fact a variant of existing algorithms, which would have a positive impact in the potential users of such technique in terms of clearly understanding their principles of working. Taking as an example the workflow of evolutionary algorithms included in Fig. 1, describing the components in a precise way would permit to determine that a metaheuristic that fits into such workflow (i.e., its main components are a kind of selection, variation and replacement strategies) can be considered as an evolutionary algorithm.

The presented use cases illustrate scenarios where our proposal demonstrates its utility applied to the process of auto-configuration of algorithms and its broader application as a comprehensive framework for integrating data that can subsequently be exported to any framework via mapping functions.

6. Conclusions

This paper proposes an ontology-based framework for standardization in multi-objective optimization, focusing on evolutionary algorithms. MOODY, the main ontology guiding the framework, covers aspects from the formalization of evolutionary algorithms and their parameters and multi-objective problems with their landscape characteristics.

MOODY is developed in OWL2 and is linked to external ontologies like BIGOWL, OPTION and DMOP. A reference implementation to ingest evolutionary-algorithms configurations is provided, converting the output of the auto-configuration tool irace to the standard format RDF.

Three use cases have been provided to validate the framework. These use cases are the enhancement of auto-configuration tools via the MOODY framework, data integration from auto-configuration tools or other sources and querying of the knowledge graph.

A line of future work is to extend the ontology by including other metaheuristics, such as particle swarm optimization, and new problems, including discrete optimization problems. In this sense, we are particularly interested in real world problems. We plan to use this semantic framework as the basis of a recommendation system to support non-expert users in finding suitable configurations for their specific problems by exploiting the similitude to known problems in their search space landscape.

CRediT authorship contribution statement

José F. Aldana-Martín: Conceptualization, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **María del Mar Roldán-García:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review &

Table 7
SPARQL Query Q4, search for algorithmic configurations that are compatible with pagmo.

```
PREFIX moody: <https://w3id.org/moody#>
PREFIX bigowl: <http://www.khaos.uma.es/lod/bigowl#>
SELECT DISTINCT ?problem_resolution ?experiment ?problem ?currentEvaluations ?hv_double
WHERE {
    VALUES ?algorithm { moody:Algorithm_NSGAII } .
    VALUES ?problem { moody:Problem_ZDT4 } .
    ?problem_resolution rdf:type moody:ProblemResolution .
    ?problem_resolution moody:partOfExperiment ?experiment .
    ?experiment moody:problemSolved ?problem .
    ?experiment moody:algorithmUsed ?algorithm .
    ?problem_resolution moody:currentNumberOfEvaluations ?currentEvaluations .
    ?problem_resolution moody:indicatorValue ?hvValue .
    ?hvValue moody:valueOfIndicator moody:QualityIndicator_HyperVolume .
    ?hvValue moody:hyperVolumeValue ?hv_double .
    ?experiment moody:parameterValue ?parameter_value_AR .
    ?parameter_value_AR moody:valueOfParameter moody:Parameter_AlgorithmResult .
    ?parameter_value_AR moody:algorithmResultValue ?value_AR .
    FILTER ( ?value_AR = "population" )
    ?experiment moody:parameterValue ?parameter_value_OPS .
    ?parameter_value_OPS moody:valueOfParameter moody:Parameter_OffspringPopulationSize .
    ?parameter_value_OPS moody:offspringPopulationSizeValue ?value_OPS .
    FILTER ( ?value_OPS = 100)
    ?experiment moody:parameterValue ?parameter_value_S .
    ?parameter_value_S moody:valueOfParameter moody:Parameter_Selection .
    ?parameter_value_S moody:selectionValue ?value_S .
    FILTER ( ?value_S = "tournament" )
    ?experiment moody:parameterValue ?parameter_value_STS .
    ?parameter_value_STS moody:valueOfParameter moody:Parameter_SelectionTournamentSize .
    ?parameter_value_STS moody:selectionTournamentSizeValue ?value_STS .
    FILTER ( ?value_STS = 2 )
    ?experiment moody:parameterValue ?parameter_value_C .
    ?parameter_value_C moody:valueOfParameter moody:Parameter_Crossover .
    ?parameter_value_C moody:crossoverValue ?value_C .
    FILTER ( ?value_C = "SBX" )
    ?experiment moody:parameterValue ?parameter_value_SCDI .
    ?parameter_value_SCDI moody:valueOfParameter moody:Parameter_SbxCrossoverDistributionIndex .
    ?parameter_value_SCDI moody:sbxCrossoverDistributionIndexValue ?value_SCDI .
    FILTER ( abs(?value_SCDI - 20.0) < 2.0)
    ?experiment moody:parameterValue ?parameter_value_M .
    ?parameter_value_M moody:valueOfParameter moody:Parameter_Mutation .
    ?parameter_value_M moody:mutationValue ?value_M .
    FILTER ( ?value_M = "polynomial" )
    ?experiment moody:parameterValue ?parameter_value_MP .
    ?parameter_value_MP moody:valueOfParameter moody:Parameter_MutationProbability .
    ?parameter_value_MP moody:mutationProbabilityValue ?value_MP .
    FILTER ( abs(?value_MP - 0.1) < 0.01)
    ?experiment moody:parameterValue ?parameter_value_PMDI .
    ?parameter_value_PMDI moody:valueOfParameter moody:Parameter_PolynomialMutationDistributionIndex .
    ?parameter_value_PMDI moody:polynomialMutationDistributionIndexValue ?value_PMDI .
    FILTER ( abs(?value_PMDI - 20.0) < 2.0)
}
ORDER BY ASC(?experiment)
```

editing. **Antonio J. Nebro:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **José F. Aldana-Montes:** Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Innovation via Grant PID2020-112540RB-C41 (AEI/FEDER, UE) and the Andalusian PAIDI program with grant P18-RT-2799. José F. Aldana-Martín is supported by Grant PRE2021-098594 (Spanish Ministry of Science, Innovation and Universities). Funding for open access charge: Universidad de Málaga / CBUA.

References

[1] M.T.M. Emmerich, A.H. Deutz, A tutorial on multiobjective optimization: fundamentals and evolutionary methods, Nat. Comput. 17 (3) (2018) 585–609.
[2] T. Weise, M. Zapf, R. Chiong, A.J. Nebro, Why Is Optimization Difficult?, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 1–50, Ch. 1.
[3] F.W. Glover, G.A. Kochenberger, Handbook of Metaheuristics, vol. 57, Springer Science & Business Media, 2006.
[4] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132.
[5] C.A.C. Coello, G.B. Lamont, D.A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, vol. 5, Springer, 2007.
[6] K. Deb, Multi-objective optimization using evolutionary algorithms, in: Wiley-Interscience Series in Systems and Optimization, 2001.
[7] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, Lect. Notes Comput. Sci. (including subseries Lect. Notes Artif. Intell. and Lect. Notes Bioinform.) 1917 (2000) 849–858.
[8] E. Zitzler, M. Laumanns, L. Thiele, Spea2: improving the strength Pareto evolutionary algorithm, TIK Rep. 103 (2001).
[9] N. Beume, B. Naujoks, M. Emmerich, Sms-emoa: multiobjective selection based on dominated hypervolume, Eur. J. Oper. Res. 181 (3) (2007) 1653–1669.
[10] Q. Zhang, H. Li, Moea/d: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[11] L.C.T. Bezerra, M. López-Ibáñez, T. Stützle, Automatic Configuration of Multi-Objective Optimizers and Multi-Objective Configuration, Springer International Publishing, Cham, 2020, pp. 69–92.
[12] S. Al-Sarayrah, D. Abulail, K. Shaalan, Understanding the Impact of the Ontology of Semantic Web in Knowledge Representation: A Systematic Review, Springer International Publishing, Cham, 2022, pp. 277–299.
[13] S. Staab, R. Studer, Handbook on Ontologies, Springer, 2010.
[14] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, J. Bouwman, A.J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C.T. Evelo, R. Finkers, A. Gonzalez-Beltran, A.J. Gray, P. Groth, C. Goble, J.S. Grethe, J. Heringa, P. A.'t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S.J. Lusher, M.E. Martone, A. Mons, A.L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M.A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The fair guiding principles for scientific data management and stewardship, Sci. Data 3 (1) (2016) 160018.
[15] I. Horrocks, U. Sattler, S. Tobies, Practical reasoning for very expressive description logics, Log. J. IGPL 8 (2000) 239–263.
[16] C. Wu, S. Liu, Z. Zeng, M. Chen, A. Alhudhaif, X. Tang, F. Alenezi, N. Alnaim, X. Peng, Knowledge graph-based multi-context-aware recommendation algorithm, Inf. Sci. 595 (2022) 179–194, https://doi.org/10.1016/j.ins.2022.02.054.
[17] F. Biscani, D. Izzo, A parallel global multiobjective framework for optimization: pagmo, J. Open Sour. Softw. 5 (53) (2020) 2338, https://doi.org/10.21105/joss.02338.
[18] T.R. Gruber, A translation approach to portable ontologies, Knowl. Acquis. 5 (2) (1993) 199–220.
[19] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, USA, 2003.
[20] P. Hitzler, J. Lehmann, A. Polleres, Logics for the semantic web, in: J.H. Siekmann (Ed.), Computational Logic, in: Handbook of the History of Logic, vol. 9, North-Holland, 2014, pp. 679–710.
[21] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, D. Tsarkov, OWL rules: a proposal and prototype implementation, Web Semant. Sci. Serv. Agents World Wide Web 3 (1) (2005) 23–40.
[22] L. Li, I. Yevseyeva, V. Basto-Fernandes, H. Trautmann, N. Jing, M. Emmerich, An ontology of preference-based multiobjective metaheuristics, arXiv:1609.08082, 2016.
[23] L. Li, I. Yevseyeva, V. Basto-Fernandes, H. Trautmann, N. Jing, M. Emmerich, Building and using an ontology of preference-based multiobjective evolutionary algorithms, in: H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, C. Grimme (Eds.), Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2017, pp. 406–421.
[24] C. Barba-González, A.J. Nebro, J. García-Nieto, M. del Mar Roldán-García, I. Navas-Delgado, J.F. Aldana-Montes, Injecting domain knowledge in multi-objective optimization problems: a semantic approach, Comput. Stand. Interfaces 78 (2021) 103546.
[25] J. Qi, L. Ding, S. Lim, A decision-making framework to support urban heat mitigation by local governments, Resour. Conserv. Recycl. 184 (2022).
[26] J. Liu, Y. Dong, Z. Liu, D. Chen, Applying ontology learning and multi-objective ant colony optimization method for focused crawling to meteorological disasters domain knowledge, Expert Syst. Appl. 198 (2022).
[27] L. Caneparo, Semantic knowledge in generation of 3d layouts for decision-making, Autom. Constr. 134 (2022) 104012.
[28] A. Kostovska, D. Vermetten, C. Doerr, S. Džeroski, P. Panov, T. Eftimov, Option: optimization algorithm benchmarking ontology, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO'21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 239–240.
[29] N.F. Noy, D.L. McGuinness, Ontology development 101: a guide to creating your first ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.
[30] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, moea/d and nsga-ii, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.
[31] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable Test Problems for Evolutionary Multiobjective Optimization, Springer London, London, 2005, pp. 105–145.
[32] F. Gu, H.-L. Liu, K.C. Tan, A multiobjective evolutionary algorithm using dynamic weight design method, Int. J. Innov. Comput. Inf. Control 8 (5 B) (2012) 3677–3688.
[33] R. Tanabe, H. Ishibuchi, An easy-to-use real-world multi-objective optimization problem suite, Appl. Soft Comput. 89 (2020) 106078.
[34] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the cec 2009 special session and competition, special session on performance assessment of multi-objective optimization algorithms, technical report 264, University of Essex/Nanyang Technological University, Colchester, UK/Singapore, 2008, pp. 1–30.
[35] S. Huband, L. Barone, L. While, P. Hingston, A scalable multi-objective test problem toolkit, in: C.A. Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 280–295.
[36] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evol. Comput. 8 (2) (2000) 173–195.
[37] C.M. Keet, A. Ławrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, M. Hilario, The data mining optimization ontology, J. Web Semant. 32 (2015) 43–53.
[38] C. Barba-González, J. García-Nieto, M. del Mar Roldán-García, I. Navas-Delgado, A.J. Nebro, J.F. Aldana-Montes, Bigowl: knowledge centered big data analytics, Expert Syst. Appl. 115 (2019) 543–556.
[39] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz Pellet, A practical owl-dl reasoner, J. Web Semant. 5 (2) (2007) 51–53, software Engineering and the Semantic Web.
[40] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, T. Stützle, The irace package: iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58.
[41] A. Liefooghe, S. Verel, B. Lacroix, A.-C. Zăvoianu, J. McCall, Landscape features and automated algorithm selection for multi-objective interpolated continuous optimisation problems, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 421–429.
[42] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.

[43] C.A. Coello Coello, M. Reyes Sierra, A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm, in: R. Monroy, G. Arroyo-Figueroa, L.E. Sucar, H. Sossa (Eds.), MICAI 2004: Advances in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 688–697.
[44] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: A. Gaspar-Cunha, C. Henggeler Antunes, C.C. Coello (Eds.), Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2015, pp. 110–125.
[45] J.D. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Report, 2006-02, Zurich.
[46] A.J. Nebro, M. López-Ibáñez, J. García-Nieto, C.A. Coello Coello, On the automatic design of multi-objective particle swarm optimizers: experimentation and analysis, Swarm Intell. (2023) 1–35.
[47] A.J. Nebro, M. López-Ibáñez, C. Barba-González, J. García-Nieto, Automatic configuration of nsga-ii with jmetal and irace, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1374–1381.
[48] K. Sörensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18.
[49] C.L. Camacho-Villalón, M. Dorigo, T. Stützle, The intelligent water drops algorithm: why it cannot be considered a novel algorithm, Swarm Intell. 13 (2019) 173–192.
[50] K. Zhang, Y. Song, A new multi-objective optimization algorithm based on combined swarm intelligence and Monte Carlo simulation, Inf. Sci. 610 (2022) 759–776, https://doi.org/10.1016/j.ins.2022.08.035.