sustainability

MDPI

Article

Ontology Design for Solving Computationally-Intensive Problems on Heterogeneous Architectures

Hossam M. Faheem [1,†], Birgitta König-Ries [2,†], Muhammad Ahtisham Aslam [3,*,†], Naif Radi Aljohani [3,†] and Iyad Katib [3,†]

[1] Computer Systems Department, Faculty of Computer and Information Sciences, Ain Shams University, Abbassia 11566, Cairo, Egypt; hmfaheem@cis.asu.edu.eg
[2] Heinz-Nixdorf Endowed Chair for Distributed Information Systems, Friedrich-Schiller-Universitat Jena, 07743 Jena, Germany; birgitta.koenig-ries@uni-jena.de
[3] Faculty of Computing and Information Technology, King Abdulaziz University, 21589 Jeddah, Saudi Arabia; nraljohani@kau.edu.sa (N.R.A.); iakatib@kau.edu.sa (I.K.)
* Correspondence: maaslam@kau.edu.sa; Tel.: +966-56-332-1977
† These authors contributed equally to this work.

Received: 13 November 2017; Accepted: 15 December 2017; Published: 8 February 2018

Abstract: Viewing a computationally-intensive problem as a self-contained challenge with its own hardware, software and scheduling strategies is an approach that should be investigated. We might suggest assigning heterogeneous hardware architectures to solve a problem, while parallel computing paradigms may play an important role in writing efficient code to solve the problem; moreover, the scheduling strategies may be examined as a possible solution. Depending on the problem complexity, finding the best possible solution using an integrated infrastructure of hardware, software and scheduling strategy can be a complex job. Developing and using ontologies and reasoning techniques play a significant role in reducing the complexity of identifying the components of such integrated infrastructures. Undertaking reasoning and inferencing regarding the domain concepts can help to find the best possible solution through a combination of hardware, software and scheduling strategies. In this paper, we present an ontology and show how we can use it to solve computationally-intensive problems from various domains. As a potential use for the idea, we present examples from the bioinformatics domain. Validation by using problems from the Elastic Optical Network domain has demonstrated the flexibility of the suggested ontology and its suitability for use with any other computationally-intensive problem domain.

Keywords: ontology design; knowledge management; heterogeneous architectures; Big Data

1. Introduction

Solving computationally-intensive problems is an attractive topic for researchers in the field of parallel processing and high-performance computing. In parallel to this, ontologies can play vital role in solving domain specific problems by making use of knowledge representation and reasoning [1–4]. Due to the nature of existing hardware clusters [5], it is now common to have a cluster of hardware architectures [6] equipped with an NVIVIA GPGPUs and Xeon-Phi coprocessor in addition to traditional Intel CPUs. Task scheduling on such heterogeneous architectures is considered to constitute a major challenge and is one of the most important topics in current scientific research. The major factors that contribute to the intensity of an integrated architecture are the complexity of modern hardware architectures, and the parallel computing paradigms and the memory management techniques associated with them [7]. Task scheduling can be characterized as the assignment of

time-constrained jobs to time-constrained resources within a pre-defined time interval, representing the complete timescale of the schedule. The domain of solving computationally-intensive problems comprises several entities, namely the jobs that solve the problem [8,9], the computing devices with the hardware architectures on which the jobs will run, the scheduling strategies used to schedule the tasks of the jobs on the hardware and the algorithms used to solve specific problems in a specific domain.

A mapping scheme from the problem domain to the computer domain should be clearly identified. For this purpose, ontologies can be developed by describing entities from one or multiple domains and relations among these entities [10,11]. Ontologies are recognized as conceptual information models that describe the entities in a specific domain, such as classes, relationships and functions [12]. In this paper, we present an ontology in the domain of High-Performance Computing (HPC) and show how an ontology-based approach may be used to solve computationally-intensive problems on heterogeneous architecture.

Many ontological reasoning-based approaches have been presented so far to address the challenges related to HPC and parallel computing. For example, a framework as an integrated solution of parallel computing and ontological reasoning is presented in [13]. It can be used on scalability issues in parallel computing by using ontological reasoning. An ontology learning-based framework to address scalability issues in parallel computing is presented in [14]. Similarly, an ontology-drive solution for cloud service handling and discovery is presented in [15,16]. In [17], the authors present an ontology editor (i.e., ONTOLIS) that can be used for concept mapping and inferencing for better content coverage in Big Data and HPC courses. In [18], the authors present a parallel ABox reasoning algorithm for increased scalability in HPC and parallel computing environments. computationally-intensive problems may be treated and solved as self-contained problems by identifying hardware, software and scheduling strategies, depending on the complexity and nature of the problem. Ontologies can play a significant role in mapping domain concepts to an inferencing environment and suggesting the best possible solutions at run time. Due to the diversity of domains and involvement of multidirectional software and hardware, very little work has been undertaken on solving computationally-intensive problems on heterogeneous architectures by designing and using ontologies.

In this paper, we present an ontology for solving computationally-intensive problems by performing reasoning on the data for given jobs. The work presented covers multidimensional domain knowledge, such as the versatility of hardware architectures, software, scheduling strategies and memory management techniques, which makes this work appropriate for domain users who belong to either the HPC or parallel computing domains. We also show the potential use of this work by presenting a real case study at the HPC Center of King Abdulaziz University.

The rest of this paper is organized as follows: Section 2 describes some work related to use of ontologies in solving various domain specific problems. Section 3 describes the structure of the ontology, including its conceptual basic building blocks. Section 4 describes the main concepts of the ontology, such as the classes, along with its attributes. Section 5 provides an evaluation to the flexibility of the proposed ontology and shows how we can easily insert computationally-intensive problem domains to it. Finally, in Section 6 we conclude our work.

2. Related Work

Ontological reasoning plays a key role in solving diverse complex problems in various domains. With the help of domain experts, semantic web experts and ontology engineers are developing ontologies that use ontological reasoning to solve problems that otherwise cannot be solved (or might not be addressed efficiently) regarding textual entities or data available in relational databases. In this section, we present work related to ontology development in various domains and the use of these ontologies in data representation, as well as solving complex problems.

An integrated solution of ontological reasoning and parallel computing to address issues of scalability is presented in [13]. Parallel computer architectures, such as home computing environments, multi-core machines, grid and peer-to-peer, lead to great demand for efficient handling

of computational resources through making use of software architecture, algorithms and now a new paradigm-making use of ontological reasoning. In [13], the authors propose using ontology modularization and queries. This helps to solve the problem of reasoning in individual modules, ultimately to achieve maximum efficiency in the use of architectures and parallel computing algorithms.

In [15], the authors present an ontology-driven platform for using and integrating the services provided by various cloud environments. The proposed framework may also be used to describe the functionality of given services by making use of ontological concepts, looking for other services from target providers to generate client adapters to use the required services. The inference engine of the proposed framework performs semantic matching to find the best possible matched service from the cloud, facilitating the resulting applications as an integrated resource of diverse services.

An ontology learning-based framework is presented in [14]. The proposed framework aims to improve scalability by making efficient use of processing power, computing resources and processing time. The proposed approach tackles the difficulties in distributed and low-level parallel programming by coupling high-level semantic descriptions with programming models.

Another approach to discover suitable services from an increasing choice on the cloud, according to user requirements such as cost, security and performance, is presented in [16]. The authors propose mapping services attributes to ontologies to minimize the gap between service descriptions, types, features and naming conventions. Using ontological reasoning on such services makes service discovery easier and more accurate on the cloud.

An ontology-based approach and framework for organizing courses covering Big Data and HPC contents is described in [17]. In this work, the authors introduce an ontology editor (i.e., ONTOLIS) that can be used to stimulate the staffing of various domains, as well as IT companies. The framework presented in [17] makes use of ontological reasoning to bridge the learning gap between domain expertise (having IT expertise; medium-level IT users; or introductory IT students). An analysis and an interpretation of archives and collections through historical data are presented in [19]. This study proposes methods for semantic associations of social networks by linking them with external datasets.

An ontology-based knowledge framework for material selection in engineering domain is presented in [20]. In this work, the authors provide a semantic representation of labeled instances, as well as the material products, representing them as RDF instances and then generating a knowledge graph from the RDF triples. The graph is generated from reasoning on the domain knowledge (modeled as classes, sub-classes, properties and individuals of the ontology). The knowledge graph generated by the knowledge framework makes suggestions for material selection.

In [21], the authors use ontological reasoning to find situations from calendar events to support users in fulfillment of their jobs. They present an ontology developed for the event domain that can be used to accommodate various situations from such events as classes and properties of the ontology. The proposed approach infers the situations by using both temporal and semantic aspects of situations. Further, the authors implemented the approach as an application for mobile phones to manage incoming calls based on the inferred situation of the user.

A parallel ABox reasoning algorithm for increasing the scalability in parallel computing environment is presented in [18]. The proposed algorithm can be used to model disjointness and inconsistencies in the ontology model, which ultimately can improve parallelization and reduce resource cost. The separation of ABox reasoning from TBox reasoning in the proposed algorithm makes the derivation simple and paralyses the integration steps, and this ultimately improves efficiency and reduces memory access.

3. Ontology Structure

The ontology's design philosophy depends mainly on its flexibility in dealing with problems from various domains. The key point is to map the algorithms used to solve problems in a given domain to equivalent algorithms in the computer science domain. Figure 1 shows the conceptual basic building blocks of an ontology. The computationally-intensive problem can be mapped to a computer algorithm

using the mapping scheme. The problem to be solved needs a computing device that contains at least a hardware architecture. The algorithm can use different parallel paradigms that run on different architectures. The scheduling strategy is responsible for assigning the appropriate architecture to the problem. In this paper, we use the bio problem as a case study of a computationally-intensive problem domain. It is clear that we can consider the proposed ontology as if it were comprised of multiple ontologies, in addition to a mapping scheme.

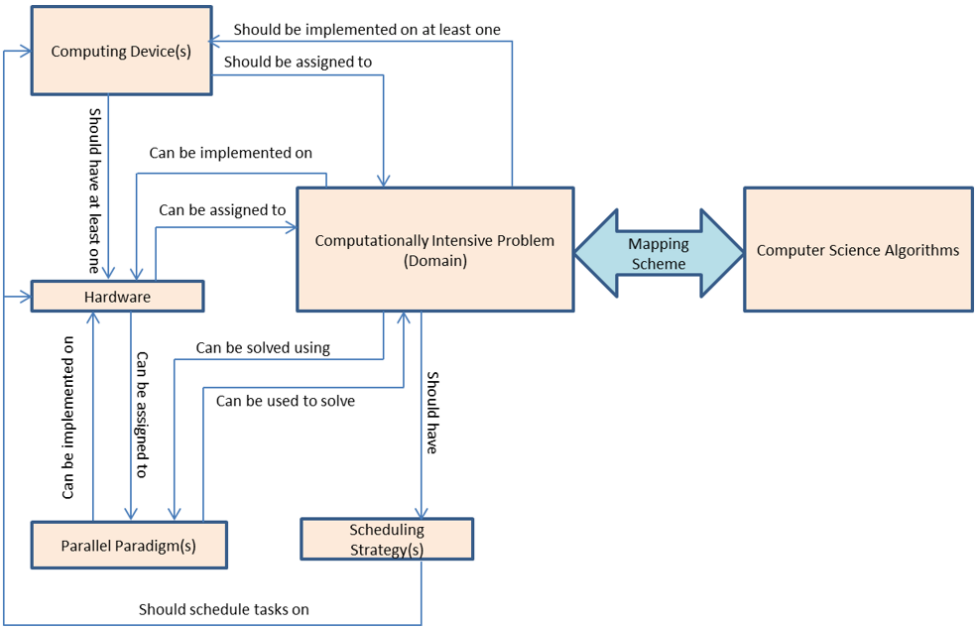

Figure 1. Conceptual Basic Building Blocks of the Ontology.

4. Ontology Basics

In this section, we describe the named classes of the ontology, the object properties and some instances. We will also outline axioms that can be used to define the meaning of several components of the ontology. Moreover, we briefly describe the mapping from the bio problem domain to the computer science domain.

4.1. Named Classes

Classes are interpreted as sets that contain individuals. They are described using formal (mathematical) descriptions that state precisely the requirements for membership of the class. The class tree contains one class called owl:Thing, which is superclass of everything. We have created five disjoint subclasses: "Algorithms", "BioProblem", "ComputingDevice", "Hardware", "ParallelParadigm", and "SchedulingStrategy".

Focusing on the BioProblem subclass can lead us to the following facts:

- BioProblem class has a set of bio problems such as "Comparing Sequences", "DNA Arrays", "Finding Signals", "Genome Rearrangements", "Identifying Proteins", "Mapping DNA", "Molecular Evolutions", "Predicting Genes", "Repeat Analysis", and "Sequencing DNA". This is shown in Figure 2. Defining these classes can help domain experts in finding the best scheduling strategy as a solution for integrated problem of bio and HPC domains.
- BioProblem class has a relation with SchedulingStrategy, Hardware, ComputingDevice, and ParallelParadigm classes, such that the BioProblem needs ComputingDevice equipped with at least hardware architecture to run the job on it. ParallelParadigm is needed to write code that best fits the selected hardware. SchedulingStrategy is used to schedule the tasks on the hardware of the ComputingDevice.

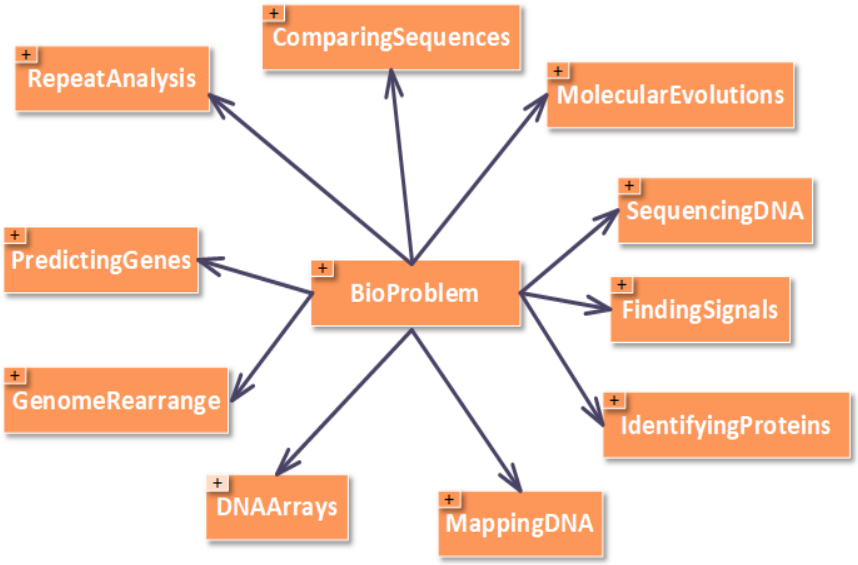

Figure 2. Bio Problem Subclasses.

The *"ParallelParadigm"* class has *MPI, OpenMP, CUDA*, and *MIC* subclasses while *"Hardware"* class has *IntelCPU, NVIDIAGPGPU*, and *Xeon-Phi* subclasses. We can see that *MPI* and *OpenMP* can work on *IntelCPU*. *CUDA* can only work on *NVIVIAGPGPU*, while *MIC* can only work on *Xeon-Phi*. *ComputingDevice* class should have at least one type of *Hardware* while it has a domain of all the hardware available. The relation between *ComputingDevice*, *Hardware*, and *ParallelParadigm* is shown in Figure 3. Ontology is designed in such a way that other parallel computing paradigms and hardware architectures can be added easily to the ontology. For example, we can add any hardware such as *ARM* which can work on *MPI* and *OpenMP*. We can also add *OpenCL* as a new *ParallelComputingParadigm* class to the *ParallelParadigm* class. Similarly, we can add *OpenCL* class that can work on *IntelCPU* and *NVIDIAGPGPU*.

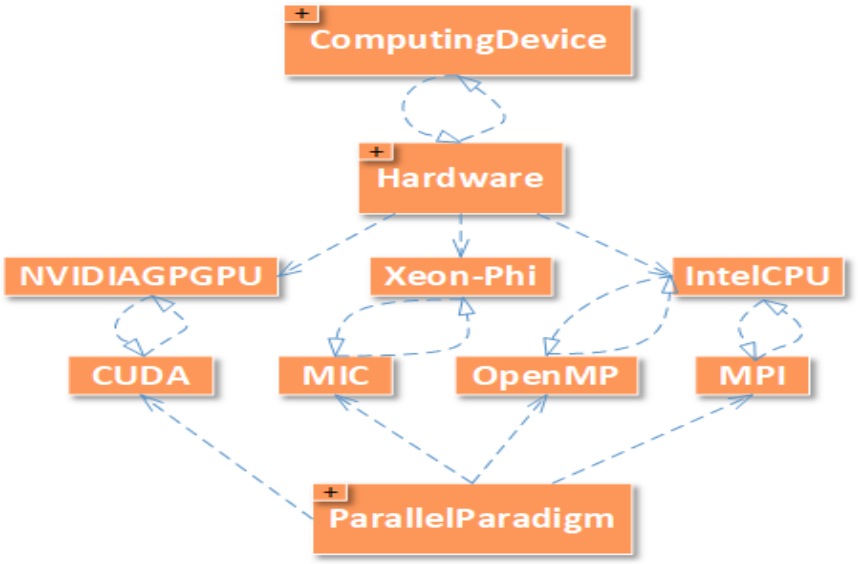

Figure 3. Relation between *ComputingDevice*, *ParallelParadigm*, and Hardware Subclasses.

4.2. Object Properties

The object feature tree contains 12 features that can be assigned to the bio problem *"hasAlgorithm"*, *"hasHardware"*, *"hasParallelParadigm"*, *"hasSchedulingStrategy"*, *"hasArchitecture"*, *"hasComputingDevice"*, *"hasParameter"*, *"IsAlgorithmOf"*, *"IsHardwareOf"*, *"IsParallelParadigmOf"*, *"IsComputingDeviceOf"*, and *"IsSchedulingStrategyOf"*.

- Property "hasAlgorithm"
  This property assigns an algorithm (from the computer science domain) to solve a given bio problem. *"IsAlgorithmOf"* is an inverse property of it.
- Property "hasHardware"
  This property assigns a hardware architecture on which the algorithm will run to solve a given bio problem. *"IsHardwareOf"* is an inverse property of it.
- Property "hasParallelParadigm"
  This property assigns a parallel paradigm used to write an algorithm to solve a given bio problem. *"IsParallelParadigmOf"* is an inverse property of it.
- Property "hasSchedulingStrategy"
  This property assigns the scheduling strategy used to deploy the hardware used to solve a given bio problem. *"IsSchedulingStrategyOf"* is the inverse of it.
- Property "hasComputingDevice"
  This property assigns at least one computing device to solve a given bio problem. *"IsComputingDevice"* is the inverse of it.
- Property "hasParameters"
  This property assigns the parameters required for each algorithm. For example, a motif-finding problem has L, d, n, and T, where: L is the length of motif, d is the permitted mutation, n is the number of characters in each sequence, and T is the number of sequences. These parameters are all of the type 'integer'.
- Property "hasArchitecture"
  This property assigns at least architecture to a computing device. Each computing device can be equipped with one or more architectures.

4.3. Instances

We initially selected two famous bio problems: "DNA sequence alignment"; and "Motif-finding Problem".

- Instance "SequenceAlignment"
  The DNA sequence alignment is one of the most famous bio problems. It is classified under "ComparingSequences". It can be solved using either combinatorial pattern matching or divide-and-conquer or dynamic programming algorithms (as shown in Figure 4).
- Instance "MotifFindingProblem"
  The motif-finding problem is one of the most famous bio problems. It is classified under "FindingSignals". It can be solved using exhaustive or greedy searches, hidden Markov models or randomized algorithms (as described in Figure 5).

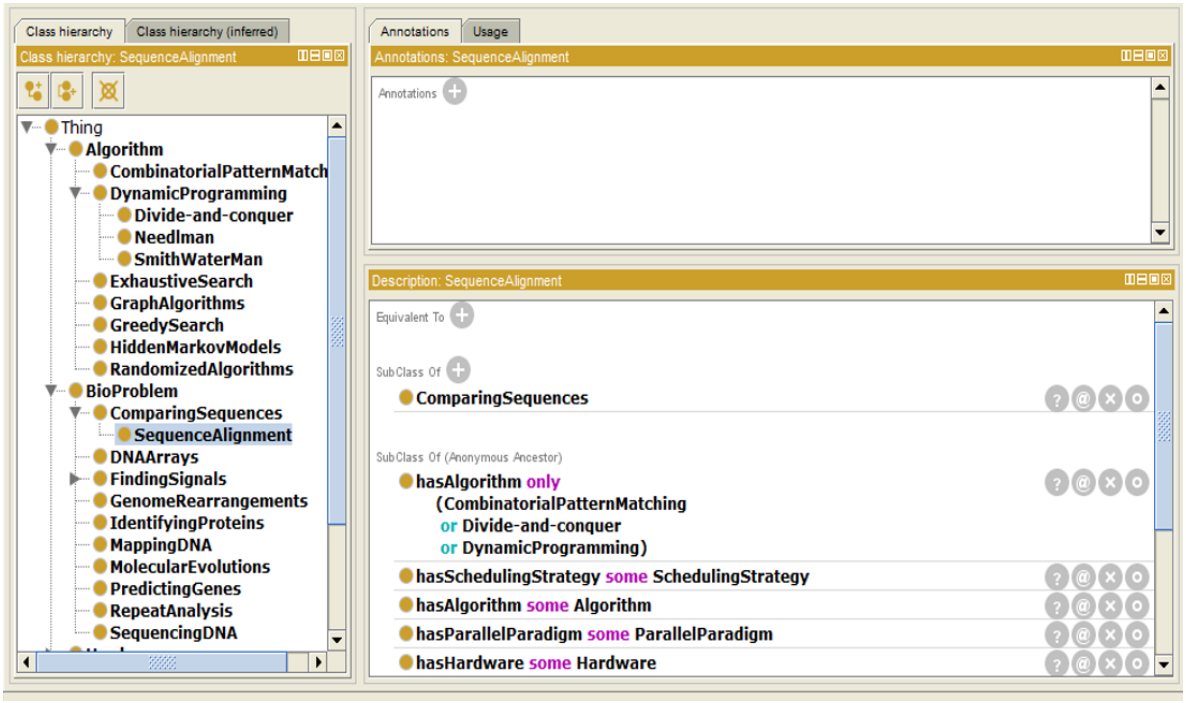

Figure 4. Instance "Sequence Alignment".

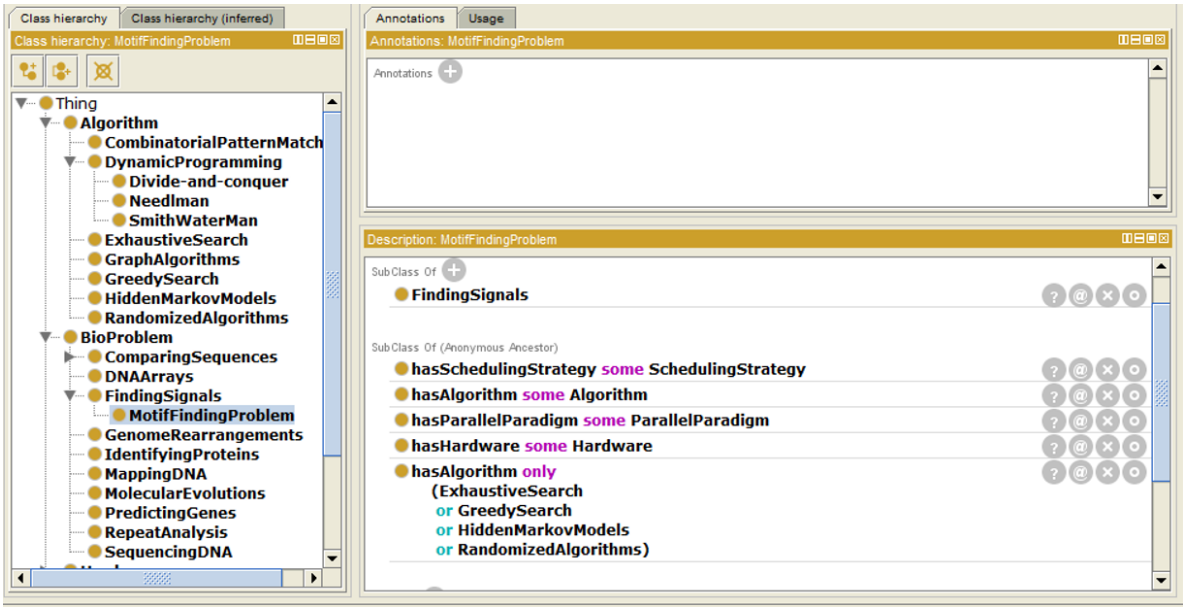

Figure 5. Instance "Motif-finding Problem".

4.4. Axioms

This section describes the axioms used to define the meaning of various components of the ontology and relationships. Table 1 lists the axioms for the ontology.

Table 1. Axioms for the ontology.

Concept Name	Axiom Description	Logical Expression
Algorithm	A collection of computer algorithms such that an algorithm is a software procedure or formula used to solve a specific problem in this domain, based on conducting a sequence of specified actions.	$A \sqsubseteq solves.Problem$
Computationally_ Intensive_Problem	A generic term to describe any problem in any domain that needs intensive computations.	$CIP \sqsubseteq Problem$
Computing_Device	A device that should contain at least one physical hardware processor.	$CD \sqsubseteq \exists contains.HardwareProcessor$
Hardware	A generic term to express the processors used in performing the computations.	$HW \sqsubseteq usedFor.Computation$
IntelCPU	A traditional type of central processing unit developed by Intel.	$ICPU \sqsubseteq CPU \sqcap madeBy.Intel$
NVIDIAGPGPU	A general-purpose graphics processor unit developed by NVIDIA.	$NGPU \sqsubseteq GPU \sqcap madeBy.NVidia$
Xeon-Phi	Term for many core processors or coprocessors developed by Intel.	$XPhi \sqsubseteq hasManyCores.ICPU$
ParallelParadigm	A paradigm used to parallelize sequential code.	$PP \sqsubseteq parallelize.SeuqentialCode$
MPI	A Message Passing Interface parallel computing paradigm that supports distributed memory multiprocessing.	$MPI \sqsubseteq PP \sqcap supports.Distributed - Memory$
OpenMP	An Open Multi-Processing parallel computing paradigm that supports shared memory multiprocessing.	$OMP \sqsubseteq PP \sqcap supports.SharedMemory$
CUDA	A Compute Unified Device Architecture parallel computing paradigm used on NVIDIA GPGPUs.	$CUDA \sqsubseteq PP \sqcap uses.NGPU$
MIC	A Many Integrated Core Architecture parallel computing paradigm used on Xeon-Phi coprocessors.	$MIC \sqsubseteq PP \sqcap uses.XPhi$
SchedulingStrategy	A scheduling strategy is used to assign suitable hardware resources to perform the jobs in a way to achieve a certain goal such as speeding up, load balancing, or optimization of power consumption, etc.	$SS \sqsubseteq assigns.HW \sqcap perfoms.Jobs$
FIFO	A First-In-First-Out scheduling strategy.	$FIFO \sqsubseteq SS \sqcap uses.FirstInFirstOut$
HEFT	A Heterogeneous Earliest Finish Time scheduling strategy.	$HEFT \sqsubseteq SS \sqcap uses.EearliestFinishTime$
PEFT	A Predictable Earliest Finish Time scheduling strategy.	$PEFT \sqsubseteq SS \sqcap uses.PredictableEarlies - tFinishTime$
Speed-based	A scheduling strategy that assigns data sets to be processed based on the speed of the processors being used to solve a problem.	$SB \sqsubseteq SS \sqcap uses.ProcessorSpeed$

5. Evaluation of the Flexibility

To evaluate the flexibility of the proposed ontology, we tested it on an additional domain called the "Elastic Optical Network (EON)". The International Telecommunication Union (ITU) divides the optical spectrum range of 1530–1565 nm (the so-called C-band) into fixed 50 GHz spectrum slots.

This fixed spectrum allocation wastes a great deal of the spectrum. EON is implemented to allow better utilization of the C-band. EON introduces plenty of challenges:

- Finding an optical path from source to destination that passes through multiple links, all of which have the same free spectrum range. This problem can be solved using either the exhaustive search algorithm, the heuristic algorithm or linear programming techniques.
- Finding a set of links that constitute an optical path, on condition that all the links have enough free contiguous spectra. This problem can be solved by using the computer science algorithm known as an exhaustive search.
- Load balancing of traffic to minimize spectrum fragmentation. This can be solved by a sorting algorithm or a binary search algorithm.

As we can see, the EON problem domain can replace the Bio Problem domain. It is also possible to create a new class called *"Computationally_Intensive_Problem"* that has several subclasses such as *"BioProblem"*, *"EON"* and so on. This can be shown as in Figure 6. The user can add as many computer algorithms as the user needs. Various scheduling strategies can be added. Eventually, it will be possible to use a ready-made ontology for computationally-intensive problems from any domain for merging with the proposed ontology.

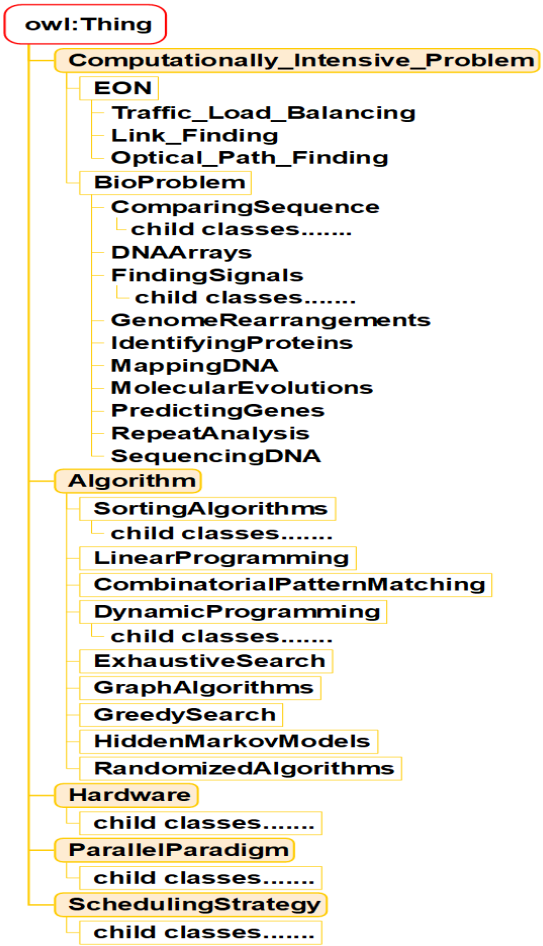

Figure 6. Class Hierarchy using *"Computationally Intensive Problem"* Class.

Now, we will describe two problems; the "motif finding problem" from the bioinformatics domain, and "Optical Path Finding" from the elastic optical network domain. The main purpose is to describe how the ontology is used to simplify and automate the whole process of problem solving.

5.1. Motif Finding Problem

The Motif Finding Problem (MFP) can be simply considered as a string matching problem. Solving the MFP to find a motif of length L with permitted mutation d can be implemented using a brute-force algorithm. All the possible *L-mers* (4^L) are compared with each possible motif of length *L*. If we have a sequence of size *N* then we can have ($N - L + 1$) motifs. In this paper, we present a problem in which the motif has a length $L = 16$, allowed mutations $d = 4$, and the number of sequences we are searching in is $T = 20$ each of size $N = 600$. All these parameters are described in the ontology. Inputs of this problem could be text files or specific DNA databases. The proposed ontology was used to describe the following:

- Computing device and its hardware used in solving the problem (*IntelCPU* and *NVIDIAGPGPU* and *Xeon-Phi*)
- Parallel computing paradigm used on each architecture (*MPI* and *OpenMP* on *IntelCPU*, *CUDA* on *NVIDIAGPGPU*, and *MIC* on *Xeon-Phi*)
- Scheduling strategy used to solve the problem (Speed-based)
- Computer algorithm used to solve the problem (exhaustive search)

We used SPARQL queries to extract all the required components from the knowledge base to implement the problem in an efficient way. The ontology was part of a large system used to solve the problem. This system includes computing cluster consisting of heterogeneous architectures, smart scheduling strategy, queuing system (PBS Pro), and algorithms library. Of course, we obtained the same results achieved when we previously deployed customized codes in [22] but in this case, the complete cycle was automated and the PBS scripts are automatically generated. This is obvious because we didn't change the algorithms used to solve the problem on the same hardware. The main concern here was how to use the ontology to simplify and automate the implementation.

5.2. Optical Path Finding

Optical path finding problem can be considered "routing and spectrum assignment problem" (RSA) which depends mainly on both the traffic demand bit-rate and the distance between the source and the destination. Routing and Spectrum Assignment (RSA) problem in a mesh network can be defined as in (1) and (2).

$$G = (v, A) \tag{1}$$

$$T = [T_{sd}] \tag{2}$$

where:

- ***G*** is a directed graph
- ***V*** is the set of nodes in the graph
- ***A*** is the set of unidirectional arcs connecting the graph nodes
- ***T*** is the traffic demand matrix

All these items are included in the ontology part related to the EON domain. Each traffic demand from a source node ***s*** to a destination node ***d*** is assigned an Elastic Optical Path (EOP) which is a physical path and contiguous spectrum based on the RSA algorithm. The assigned EOP is selected to minimize the total required spectrum used on any link. This is according to the following three conditions:

- Each demand is assigned a contiguous spectrum (spectrum contiguity constraint)
- Each demand is assigned the same spectrum across all links of its path (spectrum continuity constraint)
- Demands that share the same link are assigned non-overlapping spectrum parts

The proposed ontology was used to describe:

- Computing device and its hardware used in solving the problem (*IntelCPU* and *NVIDIAGPGPU*)
- Parallel computing paradigm used on each architecture (*OpenMP* on *IntelCPU* and *CUDA* on *NVIDIAGPGPU*)
- Scheduling strategy used to solve the problem (Speed-based)
- Computer algorithm used to solve the problem (exhaustive search)

The same methodology used to deploy the ontology in solving the "motif finding problem" was used to solve the RSA problem. We achieved the same results as in [23]. The same simplicity was achieved when the ontology was deployed in this case.

5.3. A Case Study of Bioinformatics Problem

This section describes the scheme for mapping from the bioinformatics domain to the computer science domain. Bioinformatics problems can be mapped to equivalent computer algorithms. Figure 7 shows how the graph algorithms from the computer science domain can be used to solve three different sets of problems in the bioinformatics domain. These include identifying proteins, sequencing DNA and DNA arrays. Figure 8 shows that "Finding Signals"—a problem from the bioinformatics domain—can be solved using four different sets of computer algorithms. These include the greedy search, randomized algorithms, hidden Markov models and the exhaustive search. We can conclude the cross domain problem mapping as, that one computer algorithm can solve many problems (as shown in Figure 7) and one problem can be solved by using different computer algorithms (as shown in Figure 8).

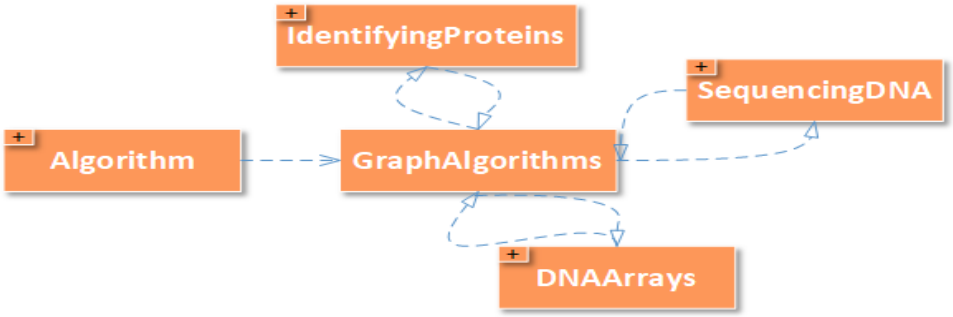

Figure 7. Use of Graph Algorithms (Computer Domain) to Solve Three Different Problems in the Bioinformatics Domain.

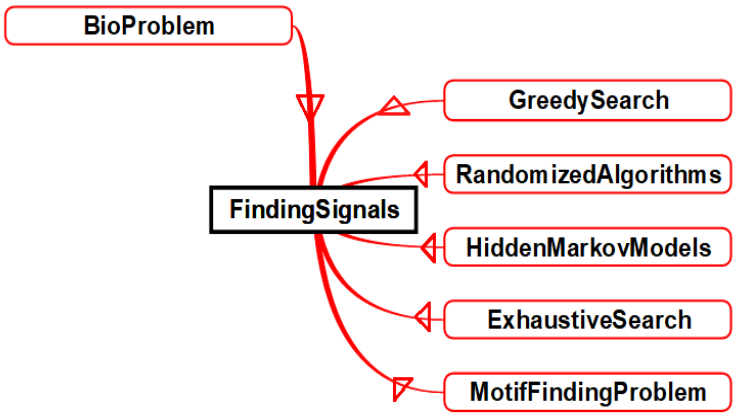

Figure 8. Solving the "Finding Signals" Problem (Bioinformatics Domain) Using Four Different Sets of Computer Algorithm.

6. Conclusions

In this paper, we presented an ontology as a semantically enriched schema for computational intensive problems. We showed how data for a domain-specific problem can be mapped to classes of the ontology and linked with each other by the object and data type properties of the ontology, so that we can perform reasoning on the given data. We also showed that computer science algorithms and hardware architectures can be flexibly appended to respond to various domain requirements. Schemes for mapping from a given domain to computer science domain are presented. To prove the usage of ontologies in solving computationally-intensive problems, we took a real-life problem from the bio domain (at HPC Center of King Abdulaziz University), mapped problem-specific data to the ontology and used the mapped data. In fact, the ability of adding new computationally-intensive problem domains is the main focus of this work to satisfy the important flexibility and reusability concepts. Flexibility is clear in the ability of the proposed ontology to add as many computationally-intensive problem domains as possible. Reusability concept is clear since the use of computer algorithms and hardware resources is common in all domains. Our case study demonstrated validation by investigating problems from two different domains.

Acknowledgments: The work described in this paper was supported by King Abdulaziz University's High Performance Computing Center (Aziz Supercomputer) (http://hpc.kau.edu.sa).

Author Contributions: Hossam M. Faheem, Birgitta König-Ries and Muhammad Ahtisham Aslam coined and worked on the feasibility of the idea. Hossam M. Faheem and Muhammad Ahtisham Aslam designed the ontology. Hossam M. Faheem, Naif Radi Aljohani and Iyad Katib designed the experiments. Hossam M. Faheem and Naif Radi Aljohani conducted the experiments in the HPC environment. Hossam M. Faheem, Muhammad Ahtisham Aslam and Naif Radi Aljohani analyzed the results and flexibility of the proposed ontology. Hossam M. Faheem and Muhammad Ahtisham Aslam wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Compton, M.; Barnaghi, P.; Bermudez, L.; Garcıa-Castro, R.; Corcho, O.; Cox, S.; Graybeal, J.; Hauswirth, M.; Henson, C.; Herzog, A.; et al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. *Web Semant. Sci. Serv. Agents World Wide Web* **2012**, *17*, 25–32.
2. Keet, C.M.; Ławrynowicz, A.; d'Amato, C.; Kalousis, A.; Nguyen, P.; Palma, R.; Stevens, R.; Hilario, M.H. The Data Mining OPtimization Ontology. *Web Semant. Sci. Serv. Agents World Wide Web* **2015**, *32*, 43–53.
3. Baker, C.; Shaban-Nejad, A.; Su, X.; Haarslev, V.; Butler, G. Semantic Web Infrastructure for Fungal Enzyme Biotechnologists. *Web Semant. Sci. Serv. Agents World Wide Web* **2006**, *4*, 168–180.
4. Shekarpour, S.; Marx, E.; Ngomo, A.C.N.; Auer, S. SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data. *Web Semant. Sci. Serv. Agents World Wide Web* **2015**, *30*, 39–51.
5. Heino, N.; Pan, J.Z. RDFS Reasoning on Massively Parallel Hardware. In Proceedings of the 11th International Semantic Web Conference (ISWC2012), Boston, MA, USA, 11–15 November 2012.
6. Atahary, T.; Taha, T.M.; Douglass, S. Hardware Accelerated Cognitively Enhanced Complex Event Processing Architecture. In Proceedings of the 2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Honolulu, HI, USA, 1–3 July 2013; pp. 283–288.
7. Miksa, T. Using ontologies for verification and validation of workflow-based experiments. *Web Semant. Sci. Serv. Agents World Wide Web* **2017**, *43*, 25–45.
8. Koumenides, C.; Alani, H.; Shadbolt, N.; Salvadores, M. Global Integration of Public Sector Information. In Proceedings of the WebSci10: Extending the Frontiers of Society On-Line, Raleigh, NC, USA, 26–27 April 2010.
9. Höchtl, J.; Reichstädter, P. Linked Open Data—A Means for Public Sector Information Management. In *Electronic Government and the Information Systems Perspective*; Andersen, K.N., Francesconi, E., Groenlund, A., van Engers, T.M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6866, pp. 330–343.

10. Bechhofer, S.K.; Stevens, R.D.; Lord, P.W. GOHSE: Ontology Driven Linking of Biology Resources. *Web Semant. Sci. Serv. Agents World Wide Web* **2006**, *4*, 155–163.
11. Jonquet, C.; LePendu, P.; Falconer, S.; Coulet, A.; Noy, N.F.; Musen, M.A.; Shah, N.H. NCBO Resource Index: Ontology-Based Search and Mining of Biomedical Resources, information retrieval, biomedical data and ontologies. *Web Semant. Sci. Serv. Agents World Wide Web* **2011**, *9*, 316–324.
12. Uschold, M. *Building Ontologies: Towards a Unified Methodology*; Technical Report; University of Edinburgh: Edinburgh, UK, 1996.
13. Bock, J. Parallel Computation Techniques for Ontology Reasoning. In *The Semantic Web—ISWC 2008, Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, 26–30 October 2008*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 901–906.
14. Arguello, M.; Gacitua, R.; Osborne, J.; Peters, S.; Ekin, P.; Sawyer, P. Skeletons and Semantic Web Descriptions to Integrate Parallel Programming into Ontology Learning Frameworks. In Proceedings of the 2009 11th International Conference on Computer Modelling and Simulation, Cambridge, UK, 25–27 March 2009; pp. 640–645.
15. Gonidis, F.; Paraskakis, I.; Simons, A.J.H. On the Role of Ontologies in the Design of Service Based Cloud Applications. In *Euro-Par 2014: Parallel Processing Workshops, Proceedings of the Euro-Par 2014 International Workshops, Porto, Portugal, 25–26 August 2014*; Revised Selected Papers; Springer International Publishing: Cham, Switzerland, 2014; Part II, pp. 1–12.
16. Ali, A.; Shamsuddin, S.; Eassa, F. Ontology-based cloud services representation. *Res. J. Appl. Sci. Eng. Technol.* **2014**, *8*, 83–94.
17. Chuprina, S. Steps towards Bridging the HPC and Computational Science Talent Gap Based on Ontology Engineering Methods. *Procedia Comput. Sci.* **2015**, *51*, 1705–1713.
18. Ren, Y.; Pan, J.Z.; Lee, K. Parallel ABox Reasoning of EL Ontologies. In Proceedings of the First Joint International Conference of Semantic Technology (JIST 2011), Hangzhou, China, 4–7 December 2011.
19. Pattuelli, M.C.; Miller, M. Semantic network edges: A human-machine approach to represent typed relations in social networks. *J. Knowl. Manag.* **2015**, *19*, 71–81.
20. Zhang, Y.; Luo, X.; Zhao, Y.; Zhang, H.C. An ontology-based knowledge framework for engineering material selection. *Adv. Eng. Inform.* **2015**, *29*, 985–1000.
21. Kabir, M.A.; Han, J.; Colman, A.; Aljohani, N.R.; Basheri, M.; Aslam, M.A. Ontological Reasoning about Situations from Calendar Events. In *On the Move to Meaningful Internet Systems: OTM 2016 Conferences, Proceedings of the Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, 24–28 October 2016*; Springer: Cham, Switzerland, 2016; pp. 810–826.
22. Faheem, H.M.; Park, S.J.; Shires, D.R. A New Scheduling Strategy for Solving the Motif Finding Problem on Heterogeneous Architectures. *Int. J. Comput. Appl.* **2014**, *101*, 27–31.
23. Fayez, M.; Katib, I.; Rouskas, G.N.; Faheem, H.M. Spectrum Assignment in Mesh Elastic Optical Networks. In Proceedings of the 2015 24th International Conference on Computer Communication and Networks (ICCCN), Las Vegas, NV, USA, 3–6 August 2015; pp. 1–6.