PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Departement

Order N°: SIOD07/M2/2021

Dissertation

Submitted to obtain the academic Master's degree in

Computer Science

Option: **Information Systems, Optimization and Decision (SIOD)**

Metaheuristics optimization for the selection of web services

By:

BENLOUANAS AMANI

Defended on 28/06/2022 before the members of the jury:

SAHLI Sihem	MAB	President
DJEROU Leila	Prof	Supervisor
HOUHOU Okba	MAA	Examiner

Academic year 2021-2022

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor Pr. DJEROU Leila for her support, guidance and tolerance; I am deeply indebted to my supervisor for her tolerance since this endeavor would not have been possible without her presence. I am honored to have had the opportunity to be taught and supervised by you. Moreover, I would like to thank the jury members for evaluating my work namely Mrs. SAHLI Sihem and Mr. HOUHOU Okba. Also, I would like to thank Pr. DJEROU Leila again for the beneficial books she offered for this research.

Dedication

To my restless parents who stayed with till the last minutes, my grandma and siblings who make my life eventful; and my aunt who motivates me. My friends who constantly help me get through rough times. Last but not least, I would like to thank myself for making for pulling through these stressful times.

All of whom I appreciate their existence and cherish. Love you!

الحمدلله

Alhamdulillah

Abstract

The web service selection is a primordial step in the support of dynamic compositions. In our master degree project, we have presented the web services selection problem as combinatorial optimization problem where the metaheuristic methods are necessary to tackle the challenge of QoS-aware web service selection. For this context, we have explored the potential of meta-heuristics for the web service selection problem. We have adapted the concepts of genetic algorithms to propose a web service selection algorithm based-GA, in web service composition. The proposed algorithm has been implemented and evaluated by some prepared examples. The evaluation results were satisfactory.

Key words: web services, combinatorial optimization, metaheuristics, service selection, quality of service.

Résumé

La sélection de services web est une étape primordiale dans le support des compositions dynamiques. Dans notre projet de master, nous avons présenté le problème de sélection de services web comme un problème d'optimisation combinatoire où les méthodes métaheuristiques sont nécessaires pour relever le défi de la sélection de services web en fonction de la qualité de service. Dans ce contexte, nous avons exploré le potentiel des méta-heuristiques pour le problème de sélection de services web. Nous avons adapté les concepts des algorithmes génétiques pour proposer un algorithme de sélection de services web basé sur le AG, dans la composition de services web. L'algorithme proposé a été implémenté et évalué par quelques exemples préparés. Les résultats de l'évaluation sont satisfaisants.

Mots clés: services web, optimisation combinatoire, méta-heuristiques, sélection de service, la qualité de service.

الملخص

يعد اختيار خدمات الويب خطوة أساسية في دعم التراكيب الديناميكية. في مشروعنا الرئيسي، قدمنا مشكلة اختيار خدمة الويب كمشكلة تحسين اندماجية حيث تكون هناك حاجة إلى طرق الأدلة العليا لمواجهة التحدي المتمثل في اختيار خدمة الويب بناءً على جودة الخدمة. في هذا السياق، استكشفنا إمكانات علم الأدلة العليا لمشكلة اختيار خدمة الويب. لقد قمنا بتكييف مفاهيم الخوارزميات الجينية لاقتراح خوارزمية اختيار خدمة الويب بناءً على خوارزمية الجينات، في تكوين خدمات الويب. تم تنفيذ الخوارزمية المقترحة وتقييمها من خلال بعض الأمثلة المعدة. نتائج التقييم مرضية.

الكلمات المفتاحية: خدمات الويب, الأدلة العليا, تحسين الاندماجية, اختيار الخدمة, جودة الخدمة.

Table of Contents

General Introduction 1

Chapter 1: Web Services

1.1 Introduction 4
1.2 Service-Oriented Architecture (SOA) 4
1.2.1 Service Oriented Architecture Roles 4
1.2.2 Service Oriented Architecture Objective 5
1.2.3 Advantage of Service-Oriented Architecture 6
1.3 Web Services 6
1.3.1 Global Definition of Web Service 6
1.3.2 Web Services Characteristics 6
1.3.3 Web Service History 8
1.3.4 Web Services Architecture 8
1.3.5 Web Services Standard Technologies 10
1.3.5.1 XML (eXtensible Markup Language) 10
1.3.5.2 SOAP (Simple Object Access Protocol) 11
1.3.6 WSDL (Web Services Description Language) 13
1.3.7 UDDI (Universal Description Discovery and Integration) 15
1.3.8 Advantages of Web Services 15
1.3.9 Drawbacks of Web Services 15
1.4 Semantic Web 16
1.5 Semantic Web Services 17
1.5.1 Framework Representation of Semantic Web Services 17
1.5.1.1 Proposed approaches for semantic web services: 17
WSDL-S (web service description language-semantic): 18
OWL-S (Ontology Web Language for Services): 18
1.6 Conclusion 18

Chapter 2: Web Services Selection

2.1 Introduction 20
2.2 Selection problem in web services 20
2.3 Quality of Service (QoS) 21
2.3.1 Definition of QoS 21
2.3.2 Key elements of QoS 22
2.4 Selection Strategies 23

2.4.1 Local Selection 23
2.4.2 Global Selection 23
2.5 Criteria for Service Selection Methods 24
2.6 Optimization problem 25
2.7 Resolution methods 26
2.7.1 Exact methods 28
2.7.2 Approximate methods 28
2.7.2.3 Single solution based 29
2.7.2.4 Population Based 29
2.7.3 Genetic Algorithms 29
2.7.3.1 The principles 30
2.7.3.2 GA Operation 31
2.8 Conclusion 31

Chapter 3: Conception

3.1 Introduction 34
3.2 Example illustration of web services selection 34
3.3 Web services selection based on meta-heuristics 35
3.4 Existing approaches for web services selection 36
3.5 Formalization of the problem 36
3.6 Genetic Algorithm 40
3.7 Service selection using genetic algorithm 42
3.8 Conclusion 43

Chapter 4: Implementation

4.1 Introduction 46
4.2.1 Development environment 46
4.2.1.1 PyCharm IDE 47
4.3.1 Generating Data 54
4.3.2 Generating Population of Bits: 55
4.4 Conclusion 57
Références 59

General Introduction

Web services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the web. An increasing number of companies and organizations only implement their core business and outsource other services over Internet. If no single web service can satisfy the functionality of user needs, there should be a possibility to combine existing service together in order to fulfill the user requirements. The process of developing such combined services is called services composition [26].

Web service composition is a process of selecting specific services from several abstract tasks and combining then into large granular services, which creates new functionalities. With the number of available web service increasing, lots of web services provide overlapping or identical functionality and different quality of services [28].

Quality of service (QoS) is an important issue in the design and management of web service composition. QoS in web services consists of various non-functional factors, such as response time, execution cost, availability, reputation and reliability.

When there are more than one candidate web services for a task or process, there will be various combinations of web services having the same functionality with different qualities. For instance, if there are m tasks and n candidate web services, the number of all possible plan is n^m. In general, finding a composition plan that fulfils a client's QoS requirement is a time-consuming optimization problem, which obliges us to adopt a meta-heuristic method, in order to select the best ones.

Meta-heuristics is an advanced algorithm to solve optimization problems. It is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions." [1]

Metaheuristic algorithms can be classified in many ways. One way is to classify them as: population-based and trajectory-based. For example, genetic algorithms are population-based that use techniques derived from genetic science and natural evolution: selection, mutation and crossover.

In our master degree project, we describe the web services selection problem which is a typical combinatorial optimization problem where we use the genetic algorithms to tackle the challenge of QoS-aware web service selection. In this context, the dissertation is structured as follow:

The first chapter provides an understanding of web services. We describe the main standards and technologies used in Web services.

The second chapter presents definitions and details about the selection problem. We provide an understanding of the principal optimization techniques.

The third chapter presents the detail of web services selection based on algorithm genetic

The fourth chapter describes the detailed design and tool used in this project, and how to implement our system web services selection based on algorithm genetic

In conclusion, we summarize and review our ideas and results by giving possible improvements and some perspectives.

Chapter 1:
Web Services

1.1 Introduction

For the last decade, the Web has evolved to encompass various information and services available worldwide. All types of organizations and companies across the world have already migrated their core activities to the Web, which has resulted in the rapid expansion of various Web applications.

This has brought about the need to build a fundamental functional infrastructure for efficient deployment and accessibility of the vast growth of Web applications. It was envisaged that the development of technologies enabling such an infrastructure would transform the business paradigm in the World Wide Web. Undoubtedly, Web services have become de facto the most significant technological by-product that caused a major transformation for the business paradigm on the Web [2].

In this chapter, we start by introducing the Web, Semantic web, then Service-Oriented Architecture, where we highlight its main realization: We describe the primary standards and technologies used in Web services.

1.2 Service-Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is an architectural approach that allows the creation of systems based on a collection of services developed in different programming languages, hosted on different platforms with different security models and business processes. Each service represents an autonomous data processing and management unit, communicating with its environment using messages. The message exchanges are organized in the form of exchange contracts. The main idea of the service-oriented architecture is that any element of the information system must become an identifiable, documented, reliable service, independent of other services, accessible, and carrying out a set of perfectly defined tasks.

1.2.1 Service Oriented Architecture Roles

Three roles make up the foundation of a service-oriented architecture.

Service provider

Web services are created by a service provider and sent to a service registry.

Service Broker

A service broker or service registry is in charge of giving a requester information about the service. It is possible for a broker to be public or private.

Service Consumer or Service Requester

A service requester locates a service through a service broker or service registry, then connects with the service provider to acquire the service.

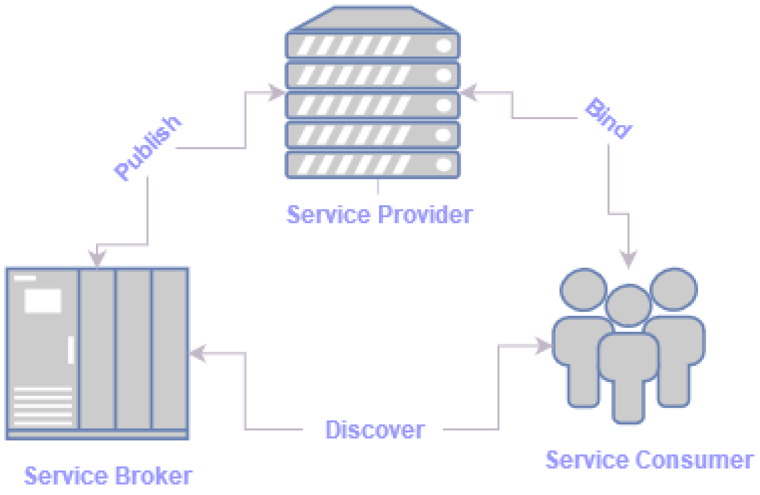


Figure 1: Service-Oriented Architecture

1.2.2 Service Oriented Architecture Objective

SOA aims to help enterprises extend the effectiveness and durability of their current IT assets, minimize architectural complexity, eliminate duplication of services and data, and boost their flexibility and agility in responding to market enterprises. SOA is a way to mold information technology around the needs of the business, instead of molding the business around it [3].

SOA has three important objectives, which are the identification of the functional components, the definition of the relationship between them, and establishing a set of constraints on each component so as to guarantee the overall properties of the architecture.

1.2.3 Advantage of Service-Oriented Architecture

The main advantage of service-oriented architecture in general and Web architecture, in particular, is that this paradigm could meet the needs in terms of flexibility and rapid adaptability expressed by designers.

1.3 Web Services

Web Services are autonomous and interdependent computational components that accomplish specified tasks ranging from basic requests to sophisticated business operations and communicate via XML messages according to the SOAP standard. They are self-descriptive, self-contained, platform-independent, publishable, and accessible via standard Internet protocols [4].

1.3.1 Global Definition of Web Service

The W3C group [5] that works on Web services has defined a Web service as

"A software system intended to support computer-computer interaction over the network. It has an interface described in a computer-processable format (e.g., WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically transmitted using the HTTP protocol and XML serialization, in conjunction with other web-related standards."

According to IBM [6]: "Web services are self-contained, self-describing, modular applications that can be published, located, and invoked from the web. Web services perform actions ranging from simple requests to complex processes. Once a web service is deployed, other applications can discover and invoke it."

1.3.2 Web Services Characteristics

Web services have characteristics that allow them to be better integrated into heterogeneous environments [7]:

Interoperability Similar to weak coupling, interoperability is a crucial criterion for Web services. Interoperability allows client applications to communicate with services that are programmed in different programming languages and running on different platforms (software or hardware). Web services rely on XML standards to simplify the construction of distributed systems and the cooperation between them.

Reusability The principle of reusability reduces development costs by encouraging the reuse of parts of code that have already been implemented. A functionality, developed in the form of a Web service, can be reused and combined with other functionalities in order to compose new services to satisfy a part of the needs. This saves a lot of time, significantly reduces the size of applications by avoiding duplication, and also facilitates maintenance.

Simplicity of use The use of standards such as XML and HTTP has highlighted the simplicity of handling Web services, as they are adapted to interconnected systems in a flexible way.

Modular Web services work in a modular and not integrated way. This means that instead of integrating all the functionalities into one global application, we create (or retrieve) several specific applications that interoperate with each other and that each performs one of these functionalities.

Weak coupling Coupling is a metric indicating the degree of interaction between two or more software components. Weak coupling is the concept that minimizes dependencies and ensures scalability, flexibility, and tolerance. This means that there is a logical separation that isolates a client application and a service in order to avoid a physical dependency between the two, i.e., the service requester does not necessarily know the provider. To do this, the client communicates with the service by exchanging messages in a standard format. This makes it possible to modify a service without breaking compatibility with existing client applications.

Autonomous and self-describing The Web services framework contains within itself all the information necessary for the use of applications, in the form of three functions: find, describe and execute in a way that is easily recognized by any external application. The ability of services to describe themselves makes it possible to consider automating the integration of services.

1.3.3 Web Service History

Web services were born out of the effort of several organizations that shared a common interest in developing and maintaining an “electronic marketplace.” They wanted to be able to communicate more simply and without having to consult on each of their transactions in order to interpret their various data. They wanted to remove the isolation of their computer systems from others [8].

By the end of the 1980s, it was clear that the age of isolated computer systems was coming to an end and that different computers of various sizes, capabilities, and shapes were emerging within the same organization. Naturally, IT departments wanted to make the best and most beneficial use of the valuable analytical power they had at their disposal. It was, therefore, necessary to make the computer applications capable of moving their work, i.e., to carry out accurate distributive processing.

Many technologies have emerged since then to respond to this new situation, always with the objective of connecting business logic through a network. This is the birth of CORBA, DCOM, Unix RPC, and Java. Still, none of these technologies has really succeeded in imposing itself because they are often attached to an operating system, an, editor or a particular language.

Today, web services are attracting a lot of interest from architects and decision-makers. Web services have already moved from the field of business-to-business exchanges to that of referencing and making company resources available, thereby infringing on EAI-type technologies. This use alone proves the quality of the model and its durability, especially at the lowest layers. On the other hand, the complete standardization of a distributed architecture built on Web Services is not yet fully established.

1.3.4 Web Services Architecture

The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

Web service architecture interacts among three roles: service provider, service requester, and service registry. The interaction involves the three operations: publish, find, and bind.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation [9].

The following figure illustrates the operations, roles, and their interaction:

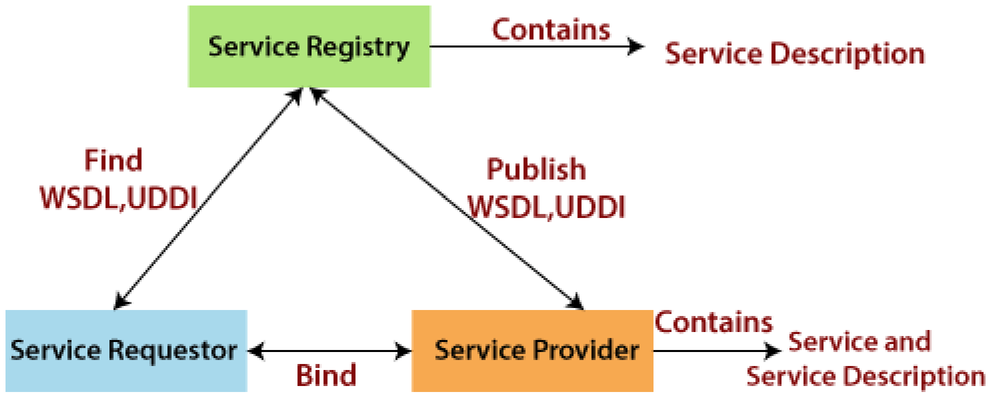


Figure 2: Web Service Architecture [10]

Web Services uses some protocol stacks that are used to define, locate, implement, and make web services interact with each other:

Transport

The transport functional component specifies the communication formats and protocols between clients and services. SOAP is a lightweight, flexible XML protocol that defines the WSA formats and protocols. SOAP offers a straightforward communications architecture that enables one program to deliver an XML message to another [7].

Description

The description functional component specifies the language used to describe a service. The service consumer binds the client to the service using the description. The WSA description language is Web Services Description Language (WSDL), an XML-based collection of definitions. A WSDL document specifies a Web service's functionality, communication protocol, and location. The different components of a Web service description may be divided into numerous documents in order to maximize flexibility and reuse. The three sections of a WSDL description are mapped to their respective WSDL definition components in Figure 3. A WSDL implementation document may be constructed to build a client proxy capable of calling a Web service through SOAP [7].

Discovery

The discovery functional component offers a registration and discovery function for services. Some discovery methods are used during development, whilst others are utilized at runtime. The WSA discovery technique is implemented using a registry service for Universal Description, Discovery, and Integration (UDDI). UDDI is mainly utilized during development; however; it may also be used during runtime. UDDI is a Web service, and SOAP messages are used to connect with it. UDDI maintains information on service kinds and service providers, as well as categorizing, locating, and binding methods for services [7].

1.3.5 Web Services Standard Technologies

The term Web Services groups together a set of XML-based technologies that make it possible to create distributed software components describe their interfaces, and use them independently of the chosen implementation language and hosting platform. SOAP, UDDI, and WSDL are the technologies that make the construction and publication of such services possible.

1.3.5.1 XML (eXtensible Markup Language)

XML is a language that allows data representation and structured documents without the use of predefined tags. Thus, this language can be extended to add specialized tags in order to describe each type of data best. This flexibility gives XML the popularity it enjoys in overcoming technical barriers.

XML is a standard that allows the description of structured documents that can be transported over Internet protocols. Indeed, it brings extensibility and neutrality to the architecture of Web services with respect to platforms and development languages.

Web services technology was designed to work in completely heterogeneous environments. However, interoperability between heterogeneous systems requires powerful mechanisms for mapping and managing message data types between different participants (clients and providers). This is a task where XML data type schemas are well suited. It is for this reason that the technology of Web services is essentially based on XML as well as the various specifications that revolve around it (namespaces, XML schemas, and Type schemas) [7].

1.3.5.2 SOAP (Simple Object Access Protocol)

SOAP, also called Service Oriented Access Protocol, SOAP is a lightweight protocol designed to facilitate the exchange of structured data in a decentralized and distributed environment. SOAP utilizes XML technologies to design an extensible messaging framework, which offers a message structure that can be sent across a number of underlying protocols. The framework was created to be agnostic of any certain programming paradigm and other implementation specific semantics [11]. It is considered to be the most important Web Services technology. The SOAP standard has been proposed to the W3C by Microsoft, IBM, Lotus. [12]

SOAP ensures interoperability between components while remaining independent of operating systems and programming languages, so theoretically, the clients and servers of these dialogues can run on any platform and be written in any language as long as they can formulate and understand SOAP messages. It, therefore, represents a basic component for developing distributed applications, which exploit functionalities published as services by intranets or the Internet [13].

Unlike other protocols, IIOP for CORBA, ORPC (Object RPC) for DCOM, or JRMP (Java Remote Method Protocol) for RMI, which are binary protocols, SOAP is based on XML to encode data. The messages exchanged via this protocol thus enjoy the advantages of the XML language to structure the data.

Structure of a SOAP message

The grammar of SOAP is quite simple to understand. It provides a means of accessing objects through remote method calls. The two strongest features of SOAP are its simplicity and the fact that everyone has accepted to use it. A SOAP message is composed of two mandatory parts: the SOAP envelope and the SOAP body, and an optional part: the SOAP header [14].

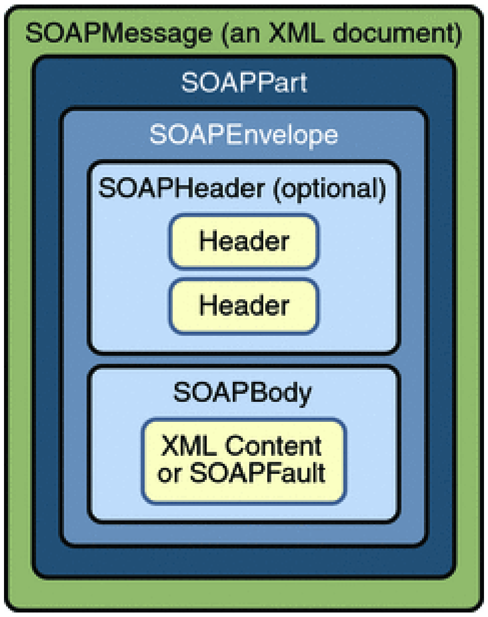


Figure 3: SOAP Message Structure [15]

1. **The SOAP envelope** serves as a container for the other elements of the SOAP message, it is defined at the beginning by the <soap:Envelope> tag and ends with the </soap:Envelope> tag [15]. SOAP messages cannot be sent in batches, in other words the envelope contains a single message consisting of an optional header (SOAP header) and a mandatory body (SOAP body) [16].

2. **The SOAP Header** is an optional part that allows to add functionalities to a SOAP message in a decentralized way without agreement between the communicating parties. It is here that it is indicated whether the message is mandatory or optional. The header is useful especially when the message must be processed by several intermediaries).

3. **The SOAP body** This is the block that contains the body of the message. It must be uniquely present in each message and be contained in the envelope. This block contains the data transported by the SOAP message which must have all its sub-elements correctly qualified by namespaces. It must contain, in sending, the name of the method called, as well as the parameters applied to this method. In response, it contains either a method call, or a one-way response, or finally a detailed error message.
4. **SOAP fault** is an optional element defined in the SOAP body and is used to report errors.

1.3.6 WSDL (Web Services Description Language)

The WSDL is a language that allows to describe Web services, and in particular, the interfaces and protocols related to Web services in XML format, and hides all the details of the Web service implementation, this will allow the requester of a service to understand the data that will be exchanged during the communication between applications and discover how a Web service can be invoked independently of any platform or programming language [25].

The WSDL allowing to provide the necessary specifications for the use of a Web service by describing:

- The methods and corresponding parameters
- The answers provided in return.
- Service access points (communication and transport protocols used).
- The input and output data formats that can be processed.
- The set of operations and messages that can be transmitted to and from a given web service.
- The location of the service where it can be invoked (URI).

A WSDL description of a Web service is made on two levels, abstract level and concrete level:

- The abstract description: It defines in an abstract way the interface of the service, i.e. the information that is independent of an implementation context.

    - ***Definitions*** This element contains the service definition. It is the root of any WSDL document. It defines the name of the web service and contains all the elements that describe the web service.
    - ***Types*** Definition of the data types used in the exchange in the form of an XML schema.
    - ***Message*** Allows the definition of data structures and their content that will be exchanged in the use of the service (data to be transmitted, return value).
    - ***Porttype*** It defines the service interface that allows the abstract definition of a set of operations offered by a Web service. Each operation refers to an input message and output messages. There are four types of operations: one-way, request-response, solicit-response and notification.
    - ***Operations*** Is an abstract description of a function supported by the Web service. It is the description of an action exposed in the port.

- The concrete description: It designates the definition of the implementation of a given service, i.e., the information related to a contextual use of the service.

    - ***Binding:*** Specifies a binding of a <porttype> to a concrete protocol (SOAP, HTTP, MIME, . . .). A <porttype> may have multiple bindings.
    - ***Service:*** This element specifies additional information needed to invoke the service. It contains several 'ports' elements, each containing a name, an access point URL and a reference to a given link.
    - ***Port:*** Network address of a link defining the access point associated with a service.

1.3.7 UDDI (Universal Description Discovery and Integration)

UDDI, was developed in 2000 as an industry initiative (Ariba, IBM, Microsoft) to become the standard registry for Web services technology. To be suitable for Web services technology, the services referenced in UDDI are accessible via the SOAP communication protocol, and the publication of information about providers and services must be specified in XML so that they can be dynamically and automatically searched and used. UDDI is a specification defining how to publish and discover Web services over a network.

1.3.8 Advantages of Web Services

The success of web services in the industry is no accident. Indeed, web services offer a number of advantages.

- **Interoperability** Web services can be used by other applications regardless of the operating system or programming languages in which they are implemented.
- **Easy and fast deployment** A company using web services can make new services available while limiting the costs and time needed to deploy them. A company can easily create a new service by combining existing services.
- **Open protocols and standards** Web services use protocols and standards that are open. Since the data and protocols are in text format, they are more easily understood and readable by developers.
- **The use of the HTTP protocol** Web services using the HTTP protocol benefit from a significant advantage, namely their operation through many firewalls without having to modify the filtering rules.

1.3.9 Drawbacks of Web Services

The web services technology also has a weak side relating to several drawbacks:

Performance issues Web services are still relatively weak compared to other distributed computing approaches such as CORBA or RMI

Security problems It is easy to bypass the security measures put in place by firewalls – HTTP doesn't have all advantages -

Availability Web services may satisfy one or more of a customer's needs, but will they always remain available and usable?

1.4 Semantic Web

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries [12]. It is an effort to enhance current web so that computers can process the information presented on World Wide Web, interpret and connect it, to help humans to find required knowledge [4]. Semantic web is intended to form a huge distributed knowledge-based system.

In the architecture figure 4, URI and Unicode, follows the important features of the existing World Wide Web. Unicode is a standard of encoding international character sets and it allows that all human languages can be used (written and read) on the web using one standardized form. Uniform Resource Identifier is a string of a standardized form that allows to uniquely identifying resources. A subset of URI is Uniform Resource Locator, which contains access mechanism and a network location of a document. Another subset of URI is URN that allows identifying a resource without implying its location and means of dereferencing it [17].

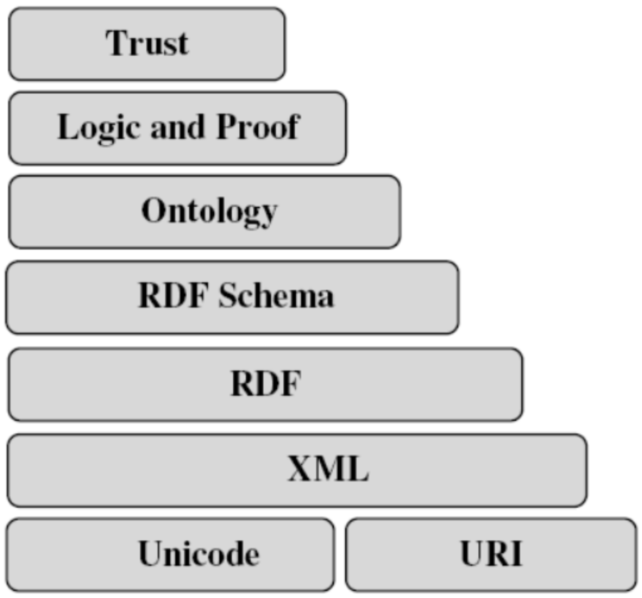


Figure 4: Semantic Web Layered Architecture [5]

1.5 Semantic Web Services

Semantic descriptions of Web services are required for their automatic discovery, creation, and execution across diverse users and domains.

Existing Web service platforms only provide syntactic descriptions, making it challenging for requesters and providers to interpret or express nontrivial claims such as the meaning of inputs and outputs or applicable limitations.

This restriction can be alleviated by supplying a comprehensive collection of semantic annotations to supplement the service description.

A Semantic Web Service is characterized by a service ontology, which enables machine interpretability of its capabilities and domain knowledge integration.

The adoption of Semantic Web Services will depend on the continued evolution and integration of Web Services and Semantic Web enabling technologies [18].

1.5.1 Framework Representation of Semantic Web Services

To work with Semantic Web Services there needs to define the relevant information reasoning problems and solutions to the problems in a consistent manner. OWL-S, WSMO, and SWSF are three major approaches for such representation frameworks that have been proposed.

The fundamental language for service ontologies in OWL-S is description logic (OWL). WSMO emphasizes loose coupling between services and aims to achieve this via distinct goal and service ontologies and an enhanced mediator architecture. OWL-S is the foundation of SWSF, but an existing ontology called Process Specification Language enhances its process model (PSL) [17].

1.5.1.1 Proposed approaches for semantic web services:

The semantic Web service cannot exist without the development of a set of universal standards and architectures, just as the current Web could not have existed without HTTP and HTML. To be able to express semantics with Web information, many languages have been developed. We will therefore present the objectives and functionalities of the main languages dedicated to semantic Web services [17]:

WSDL-S (web service description language-semantic):

WSDL-S is a semantic description language for web services. A WSDL-S web service description is a WSDL description augmented with semantics, this semantics is added in two steps: The first step is to reference, in the definition part of WSDL, an ontology dedicated to the service to be published. The second step consists in annotating the operations of the WSDL definition with semantics by adding two new tags: the Action tag and the Constraint tag [17].

OWL-S (Ontology Web Language for Services):

OWL-S is a description language and an OWL (Web Ontology Language) ontology for web services. The structuring of the OWL-S top ontology is motivated by the need to provide three types of information essential for a service, namely: What does the service do? How does the service work? How is the service accessed? [19].

1.6 Conclusion

Web services technology offers great potential for overcoming system interoperability issues which makes it an ideal technology and promising framework for the integration and interoperability of distributed systems.

In this chapter, we have presented a general vision of Service-oriented architecture, and at the center of it we defined and described the concept of Web Services, their function and technologies, at the end we learned of their advantages and drawbacks.

In the next chapter, we discuss the selection of web services and their process and the approach we used and why it was used.

Chapter 2:
Web Services Selection

2.1 Introduction

The web service technology is an implementation of the service-oriented architecture (SOA), they are based on a set of standards such as SOAP, UDDI and WSDL, which enable a flexible way for applications to interact with each other over networks. In spite of its advantages, the infrastructure is not sufficient to ensure the automation of several tasks such as the service selection. In many cases, the industrials need to compose several services (or tasks) in order to fulfill a given requirement [20]. For example, a travel booking web service can be built by aggregating a flight booking web service, a car rental web service, a travel insurance web service, an accommodation booking web service, a payment web service, and an itinerary planning web service. Each service may have many implementations, all of which have the same functionality, but have different Quality of Service (QoS) values. Thus, a significant research problem in web service composition is how to select a web service implementation for each of the web services such that the composite web services give the best overall performance [21].

This chapter describes the web services selection problem which is a typical combinatorial optimization problem where the heuristic and metaheuristic methods are necessary to tackle the challenge of QoS-aware web service selection.

2.2 Selection problem in web services

When an application is web-based, its component tasks can be performed using web services. In particular, for the same task, we may discover several web services capable of executing it; we would then face a web service selection problem to choose the combination of services that provides the best quality service. This selection must also take into account the respect of the real-time constraints imposed when it is a real-time application. The following figure describes the selection problem, which goes through several phases: service discovery, which is a search for services that match the functionality and non-functionality requirements for each task in the composition is performed. Next, from the multiple services discovered in the previous phase, service selection comes to select the most appropriate service for each task in the composition in order to satisfy user requirements. The last phase is the service execution, where the individual task in the composition is invoked and executed to come up with the final service [22].

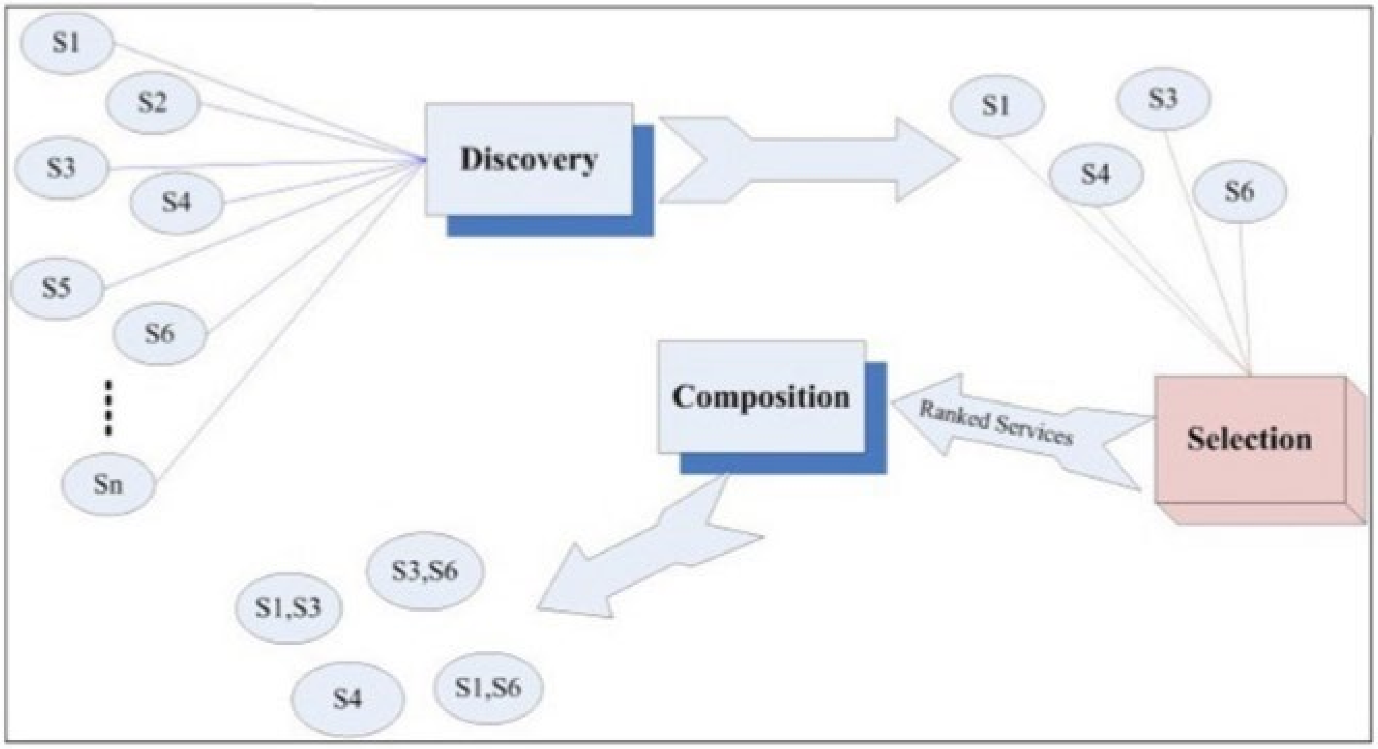


Figure 5: The Discovery, Selection, and Composition of Services

2.3 Quality of Service (QoS)

2.3.1 Definition of QoS

Quality of Service (QoS) is a set of properties and characteristics of an entity or service that gives it the ability to satisfy stated or implied needs. These needs may be related to parameters such as accessibility, availability, response time, cost, reliability, etc. QoS thus becomes an important criterion that determines the usability of the service and utility, both of which influence the popularity of a particular Web service, and an important selling and meeting point differentiation between service providers Web. In the context of Web services, QoS can be viewed as an assurance on a set of quantitative characteristics [23].

2.3.2 Key elements of QoS

The key elements of QoS support in a web services environment are summarized in the following [24]:

- **Availability:** This is the absence of service interruptions. Availability represents the probability that a service is available. Higher values mean that the service is always ready for use while lower values indicate unpredictability as to whether the service will be available at a particular time.

- **Accessibility:** Represents the degree to which a web service request is served. It can be expressed as a probability measure indicating the rate or chance of a successful service instantiation at a point in time.

- **Standards Compliance:** Describes the compliance of a Web service with standards. Strict adherence to correct versions of standards by the provider of a service is necessary for the proper calling of Web services by the requesting service. In addition, service providers must adhere to the standards set forth in the service level agreements between requesters and service providers.

- **Integrity:** Describes the degree to which a Web service performs its tasks according to its WSDL description, as well as compliance with Service-Level Agreement (SLA). A higher degree of integrity means that the functionality of a service is closer to its WSDL or SLA description.

- **Performance:** Performance is measured in terms of two factors: throughput, latency and response time. Throughput represents the number of Web service requests served in a period of time. Latency is the time between sending a request and receiving a response.

- **Response time:** The time required to complete a higher throughput and lower latency values represent the good performance of a web service.

- **Reliability:** Represents the ability of a service to operate correctly and consistently and to provide the same quality of service despite system or network failures.

- **Security:** Involves aspects such as authentication, authorization, message integrity, and confidentiality. Security has additional importance because the invocation of Web service occurs over the Internet. The amount of security a particular Web service requires is described in its accompanying SLA, and service providers must maintain that level of security.

- **Scalability:** Scalability refers to the ability to consistently serve requests despite variations in the volume of reque3sts. Highly accessible web services can be achieved by building highly scalable systems.

2.4 Selection Strategies

Several works have been done in the area of selection of Web services according to non-functional requirements. There are mainly two selection strategies: a local selection strategy and a global selection strategy. Applying one or the other strategy amounts to executing an appropriate selection algorithm.

2.4.1 Local Selection

Its objective is to select the best Web service for each individual task. In this approach the selection of the Web service that will execute a given task of a composite service specification is done at the last possible moment and without considering the other tasks involved in the composite service. When a task actually needs to be executed, the system collects QoS information from each of the Web services that can execute that task. After collecting this QoS information, a quality vector is calculated for each of the candidate web services, and based on these quality vectors, the system selects one of the candidate web services.

2.4.2 Global Selection

The goal of the global selection strategy is to select the combination of web services that guarantees the best overall quality taking into account the QoS constraints and global preferences assigned for the set of tasks. For the case of a real-time application, the goal is to select the combination of services that provides the best QoS and respects the real-time constraints and other global preferences imposed for the set of tasks of the application such as imposing that the end-to-end runtime must be less than a deadline.

2.5 Criteria for Service Selection Methods

The availability of service providers with different characteristics makes the task of choosing an appropriate service provider for a user increasingly complex which motivates us to consider new solutions for the service selection problem [27].

- There are different types of services (calculation, storage, etc.).
- There are a large number of services with similar functionalities which results in a proliferation of services that offer similar functionalities with different QoS criteria (price, availability).
- There are a large number of service providers that are emerging continuously on the Internet such as Google AppEngine.

Finally, there is a wide range between service performance and price. When different providers offer their services with different price and performance values.
It can be concluded that the process of web service selection needs five crucial outcomes [27]:

- Clarification: the algorithm should avoid the loss of web services that may match the user's request, but their interface is not the same as the user's request.
- Flexibility: new scalable mechanisms should be flexible to support a large number of service providers.
- Scalability: the selection algorithm should be scalable to support a number of QoS requirements.
- General: The selection algorithm should be as generic as possible to support different users and different user requirements, rather than specific types of users.

- User personalization: the algorithm must be able to provide the right service at the request of the users.

Consequently, the web selection service is a typical combinatorial optimization problem where the heuristic and metaheuristic methods are necessary to tackle the challenge of QoS-aware web service selection.

2.6 Optimization problem

In mathematics, computer science and economics, an optimization problem is the problem of finding the best solution from a set of feasible solutions S (also called decision space or search space), for the solution(s) x∗ that minimizes (or maximizes) a function measuring the quality of that solution.

When we want to solve an optimization problem, we look for the best possible solution to this problem, i.e., the global optimum. However, there may be intermediate solutions, which are also optimum, but only for a restricted subspace of the search space: we then speak

of local optimums. This notion is illustrated in Figure 6.

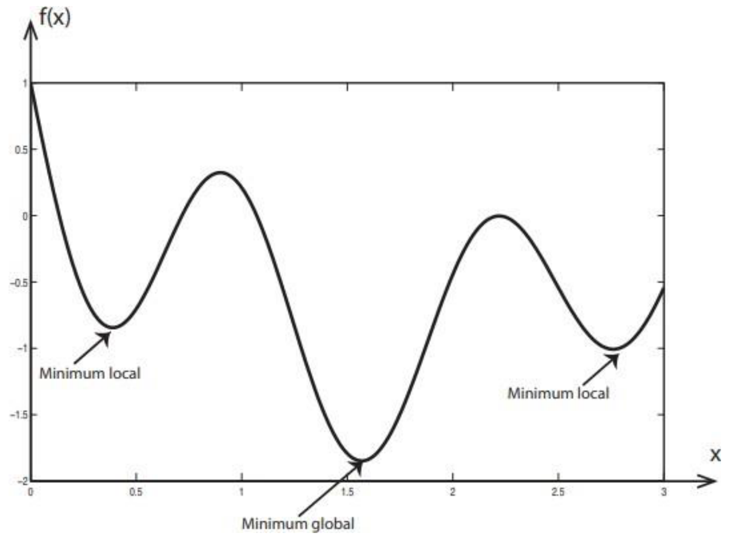


Figure 6: Global and Local Optimum

Depending on the nature of the “spaces” in which the decision variables take their values, a distinction can be made between continuous optimization and discrete (or combinatorial) optimization. For continuous optimization, the solutions are vectors of real numbers (we speak of real variables), and the space of decision variables (the search space) is infinite. However, for discrete optimization the search space is finite and discrete. Discrete problems are generally combinatorial.

A combinatorial optimization problem or NP-optimization problem (NPO) (also called discrete optimization) is an optimization problem whose feasible sets are finite but combinatorial. A combinatorial optimization problem can be defined by:

– A vector of variables $x = (x_1, x_2, \ldots, x_n)$,

– Domain of variables $D = (D_1, D_2, \ldots, D_n)$, where (D_i), $i=1, \ldots, n$ are finite sets,

– Set of constraints,

– An objective function f to be minimized or maximized,

– The set of all possible feasible solutions is $S = \{x = (x_1, x_2, \ldots, x_n) \in D / x$ satisfies all constraints$\}$, the set S is also called a search space.

Solving a combinatorial optimization problem consists of:

- Modeling a problem: search space, solutions

- Formalize mathematically: objective function, constraints

- Apply an optimization method

- Obtaining a solution

2.7 Resolution methods

There are several methods for solving a combinatorial optimization problem which are divided into two categories: exact methods and approximate methods

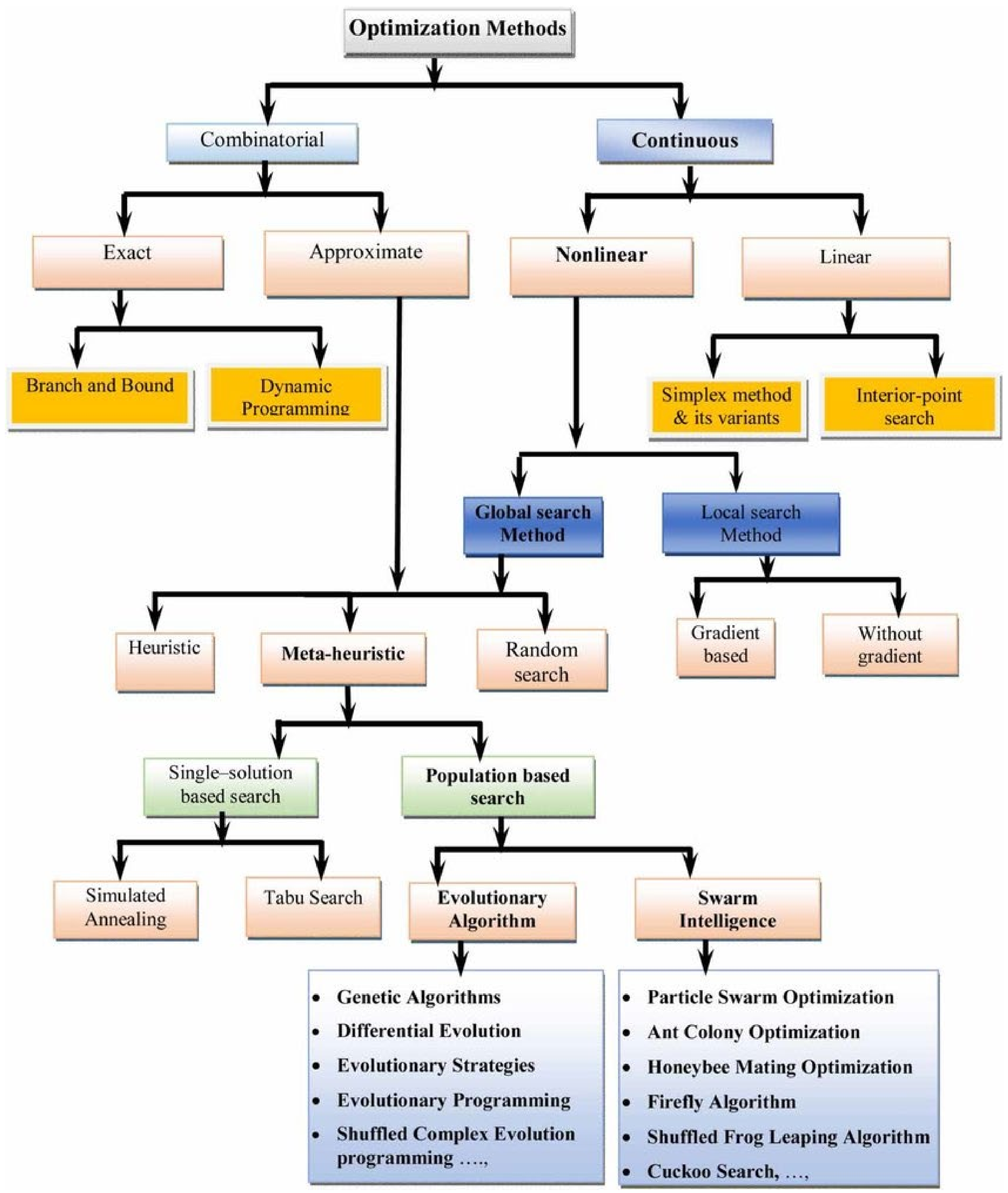


Figure 7: Optimization Methods

2.7.1 Exact methods

Solving a multi-objective optimization problem in an exact way means that we want to obtain either the whole set or a subset containing exactly one optimal solution, exact methods ensure this objective with a complete walk over the whole search space in order to remove all solutions that may be much better than the current optimal solution.

2.7.2 Approximate methods

Approximate methods do not guarantee obtaining an exact solution, but only an approximation. They explore a sub-part of the search space. Approximate methods can be heuristics or meta-heuristics, they generally exploit random processes in the exploration of the search space [1].

2.7.2.1 Heuristic Approach

A heuristic method is a method of problem-solving that does not rely on a detailed or exhaustive analysis of the problem. It consists of successive approaches based, for example, on similarities with problems already dealt with in order to gradually eliminate the alternatives and retain only a limited series of solutions to tend towards the optimal one. Heuristics take much less time to find the optimal solution compared to the Exact Method [1].

2.7.2.2 Metaheuristic approaches

Faced with the difficulties encountered by heuristics to obtain a good quality feasible solution for difficult optimization problems, meta-heuristics appeared.

They are generally iterative stochastic (randomly determined) algorithms, which progress towards a global optimum, i.e., the global extremum of a function, by sampling an objective function. They behave like search algorithms, trying to learn the characteristics of a problem in order to find an approximation of the best solution (in a way close to approximation algorithms) [1].

There are two approaches to meta-heuristic methods, single solution meta-heuristics and population of solutions meta-heuristics.

2.7.2.3 Single solution based

This category initializes the search with a unique solution and then improves it during a series of iterations based on the notion of neighborhood. The initial solution undergoes modifications according to its neighborhood in order to progressively improve its quality. We find in this class two families of local search algorithms: For a local optimization: simple local search (descent). For a global optimization: Simulated annealing.

2.7.2.4 Population Based

This category initializes the search with a set of solutions in order to obtain the best (optimal) solution which is the solution of the problem being addressed. The use of a solution set increases the possibility of arriving at a good quality solution. The categories of this class: evolutionary algorithms, genetic algorithms, and swarm intelligence-based algorithms: Ant Colonies, Fruit Fly.

2.7.3 Genetic Algorithms

Genetic algorithms (GAs) are now well-known stochastic optimization methods, inspired by the mechanisms of natural selection and genetics. They use the principles of survival of the best adapted individuals. It was [28], who laid the theoretical foundations of the genetic algorithm, moving from the Darwinian paradigm of natural evolution to that of artificial evolution. The main goal of these algorithms is to find a solution for difficult problems - problems where no exact methods are known to solve them in reasonable time for example, they are applied in various fields to solve optimization or search problems such as web service selection.

A new step was taken when the work of G. Goldberg, in the mid-eighties, gave genetic algorithms their credentials as a viable, efficient and non-specific optimization method. Some basic terms of the genetic algorithm:

- **Population:** finite set of individuals (of solution).
- **Individual:** potential solution of the problem or expression of chromosomes.
- **Chromosome:** potential solution of the problem in a coded form (string form) or set of genes.
- **Gene:** a non-divisible elementary part (character) of a chromosome.
- **Fitness:** a term which designates the evaluation function of an individual.

This function is linked to the function to be optimized and makes it possible to define the degree of performance of an individual (thus of a solution). This fitness is equal to the objective function (F) in the minimization case and (1/F) in the maximization case.

2.7.3.1 The principles

Genetic algorithms are optimization approaches that use techniques derived from genetic science and natural evolution: selection, mutation and crossover. To use these approaches, one must implement the following:

a) The coding of a population element (individual): a function that allows to model the data of the real problem in data usable by the genetic algorithm

b) A function to generate the initial population: the generation of the initial population is important since this generation represents the starting point of the algorithm and its choice influences the speed and optimality of the final solution.

c) The objective function (fitness function): a function that returns a fitness value for each individual. This value allows to determine the relevant solution since the problem is restricted to find the group of individuals who have the optimal values.

d) Operators that allow the evolution from one population to another while improving the objective function. The crossover operator recomposes the genes of existing individuals in the population, while the mutation has the goal but guarantees the exploration of the state space.

e) Dimensioning parameters: population size, total number of generations (stopping criteria), probabilities of application of crossover and mutation operators, etc.

2.7.3.2 GA Operation

The operation of a genetic algorithm is based on the following phases:

a) **Initialization:** Choice of the individuals N which represents the initial population.

b) **Evaluation:** Each individual is evaluated by the objective function.

c) **Selection:** Creation of a new population of N using selection methods (reproduction/replacement)

d) **Reproduction:** It uses genetic operators (crossing and mutation) to produce the new generation. Mutation operators modify one individual to form another while crossover operators generate one or more children from combinations of two parents.

The objective function (fitness) determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

2.8 Conclusion

In this chapter, we introduced the concepts of metaheuristics optimization, and then the evolutionary genetic algorithm as the best algorithm for solving multi-objective optimization problems.

Genetic algorithms provide solutions close to the optimal solution using selection, hybridization and mutation mechanisms. They are the most widely used metaheuristic approaches for solving difficult optimization and search problems. Their efficiency is determined by the genetic operators used and by the evaluation function.

Chapter 3:
Conception

3.1 Introduction

The web service selection is a primordial step in the support of dynamic compositions. Web service composition create new functionalities by aggregating different services based on specific workflow. When there are more than one candidate web services for a task or process, there will be various combinations of web services having the same functionality with different qualities. For instance, if there are m tasks and n candidate web services, the number of all possible plan is n^m. In general, finding a composition plan that fulfils a client's QoS requirement is a time-consuming optimization problem, which obliges us to adopt a meta-heuristic method, like genetic algorithm, in order to select the best ones.

In this chapter, we present the web service selection problem, in web service composition, and how we treat it using genetic algorithm.

3.2 Example illustration of web services selection

We assume that we have a set S of m service tasks (abstract classes of web services), $S=\{t_1, t_2, \ldots, t_m\}$. Each service class $t_j \in S$ has candidates atomic services (concrete services) $s_{j\,i}$, $\boldsymbol{i} \in [\mathbf{1}, \boldsymbol{n}]$, with the same functions Fj and different QoS. The QoS values (Time , Price, Availability) of all component services are shown in Figure, where :

Time: response time ≤600; Price : price ≤ 250 ; Availability : availability ≥85%

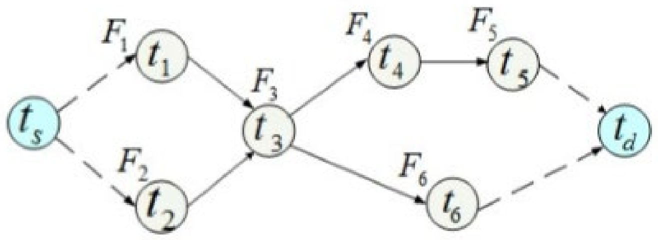
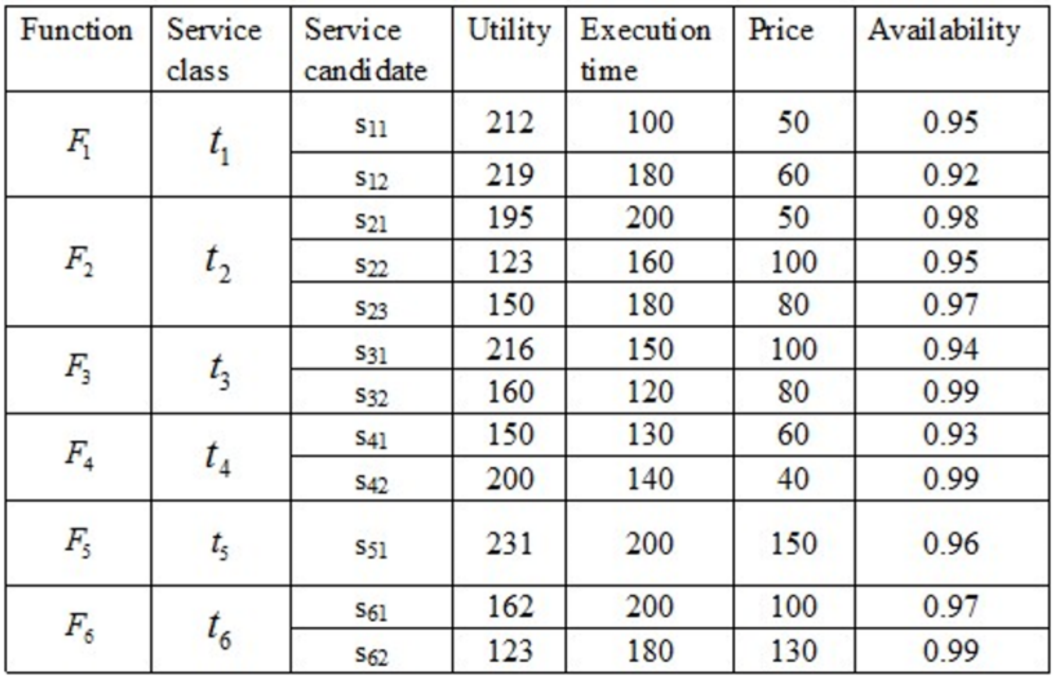


Figure 8: Composite Web Service Example [29]

Table 1: QoS Parameters [29]



Function	Service class	Service candidate	Utility	Execution time	Price	Availability
F_1	t_1	s_{11}	212	100	50	0.95
		s_{12}	219	180	60	0.92
F_2	t_2	s_{21}	195	200	50	0.98
		s_{22}	123	160	100	0.95
		s_{23}	150	180	80	0.97
F_3	t_3	s_{31}	216	150	100	0.94
		s_{32}	160	120	80	0.99
F_4	t_4	s_{41}	150	130	60	0.93
		s_{42}	200	140	40	0.99
F_5	t_5	s_{51}	231	200	150	0.96
F_6	t_6	s_{61}	162	200	100	0.97
		s_{62}	123	180	130	0.99

The optimal selection plan of services, which maximizes the sum of all service utilities, {(s_{11}, t_1), (s_{11}, t_1), (s_{11}, t_1), (s_{11}, t_1)} with a total utility of 823, a response time of 590, at a price of 240 and an availability of 86.64%

3.3 Web services selection based on meta-heuristics

Web service selection problem, in web service composition, has some proprieties that make the meta-heuristic methods as required approaches to solve it. We can summarize them as follow [29]:

NP difficulty: Web service selection and composition problem is NP, and traditional methods cannot bear the complexity burden. Correspondingly, the meta-heuristic approaches can provide satisfactory solutions in a reasonable time;

Multiple choices: Users generally have different preferences and different persons may have different opinions. Correspondingly, population-based meta-heuristic approaches can present multiple solutions.

Dynamic: Web services are executed on the internet and the composite services have great dynamic properties, for example, the selected component services are suddenly unavailable and QoS values changed. Simultaneously, users are possible to modify their choices and/or constraint conditions. At these scenarios, traditional methods maybe need to replan the composite service. Correspondingly, meta-heuristic methods are very possible to present another schema immediately at any time and in any situation due to they maintain a solution population.

3.4 Existing approaches for web services selection

There are several approaches proposed in the domain of service selection. These approaches can be grouped in two major categories: the multi-objective optimization and the mono-objective optimization [20]. The multi-objective selection is not addressed by this work.
The mono-objective category can use a global selection model or a local selection model or a hybrid selection model. The global selection model can determine the optimal solution, but it has an exponential complexity, however the local model has only a linear complexity but cannot handle the global constraints (there are only, local constraints). The third category is a compromise of the two approaches, it has a reduced complexity in comparison with the global approach, and it can also handle the global constraints.

3.5 Formalization of the problem

Let CA = {S1, . . . , Sn} be a client request (or an abstract composition) And Cons a vector of m global QOS constraints {cons1,.. , consR}. We must find a concrete composition c={s1, . . . , sn} by binding each Sj to a concrete service sj' ∈ Sj such that: 1. U'(c) is maximized, and 2. Q' k(c) ≥ Cons(k), ∀ Cons(k)∈ CONS

To formalize the QoS-based web service selection problem, we need to design an objective function that takes into account the QoS criteria, as well as the global constraints. The non-functional properties of a web service are expressed by QoS attributes:

- Response time: the time required to complete a request by a web service.

- The cost: this is the price that a customer of the service must pay to benefit from the web service.
- Availability: the probability that the service will be available at a certain time period.
- Reputation: this is a measure of the credibility of a web service.
- Reliability: the ability of a service to respond correctly to a request.

The set of QoS attributes can be divided into two categories: negative attributes and positive attributes. Positive attributes are to be maximized (e.g., availability, reliability, etc.) while negative attributes are to be minimized (e.g., response time, cost, etc.). To homogenize these criteria, we convert the negative attributes into positive attributes by multiplying their values by -1.

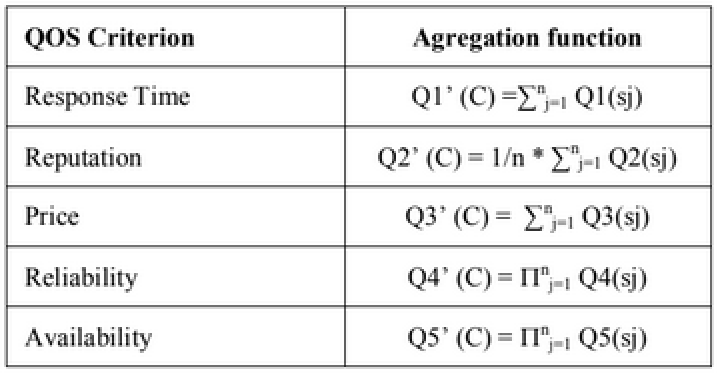
It is assumed that we have R QoS attributes, the vector $Q = \{Q_1(S), Q_2(S), ... Q_R(S)\}$ represents the QoS attributes of a service S, where $Q_i(S)$ determines the value of the i^{th} attribute of S. In our work, we used five QoS attributes: response time, cost, availability, reputation and reliability [30].

The QoS value of a composite service (composition) depends on the QoS property values of its components (atomic web services selected in the composition), as well as the workflow used. There are several basic composition structures: sequential, parallel loop (split/ join) and conditional (exclusive split/exclusive join). In our work, we consider only the sequential model.

The QoS criteria must be normalized, to avoid the problem of compensation between criteria. For this purpose, we use a number of aggregation functions.

Our model uses three types of QOS aggregation functions summation (for the price, the reputation and the response time, 2) multiplication for the reliability and availability and 3) minimum/maximum relation for the values normalization.

Table 2: Aggregation Functions [20]



QOS Criterion	Agregation function
Response Time	$Q1'(C) = \sum_{j=1}^{n} Q1(sj)$
Reputation	$Q2'(C) = 1/n * \sum_{j=1}^{n} Q2(sj)$
Price	$Q3'(C) = \sum_{j=1}^{n} Q3(sj)$
Reliability	$Q4'(C) = \prod_{j=1}^{n} Q4(sj)$
Availability	$Q5'(C) = \prod_{j=1}^{n} Q5(sj)$

Global QoS constraints can be expressed in terms of upper and/or lower bounds on the aggregate values of the various QoS properties. These bounds represent the user's requirements on the QoS properties of the composition (e.g., the total cost of the composition must be less than \$2). We consider only the positive properties to deal only with the lower bounds.

The minimum and maximum values of the kth QoS criterion of a composition c are:

$Qmin'(k) = \sum n_j=1\ Qmin(j,k)$ .... (1)

$Qmax'(k) = \sum n_j=1\ Qmax(j,k)$ .... (2)

$Qmin(j, k) = min_{\forall s_{ji} \in S_j}\ Qk(s_{ji})$.. (3)

$Qmax(j, k) = max_{\forall s_{ji} \in S_j}\ Qk(s_{ji})$ .... (4)

With:

$Qmin(j,k)$ is the minimum value of the k^{th} criterion (e.g. the minimum cost) belonging to the abstract class *j*.

$Qmax(j,k)$ is the maximum value of the k^{th} criterion (e.g. the maximum cost) belonging to the abstract class *j*.

$Qmin'(k)$ and $Qmax'(k)$ are calculated with a sum, since we consider only the sequence.

To homogenize the normalization of the QOS criteria, we multiply the negative criteria with -1, and in this way all criteria should be maximized.

The utility function of a service instance s (respectively of a composition *c*) is given as follows:

$\mathrm{U(s)} = \Sigma_{k=1}^{R} \mathrm{w}_k * (\mathrm{Q}_k\ (\mathrm{s}) - \mathrm{Qmin(j,k)})/(\mathrm{Qmax(j,k)} - \mathrm{Qmin(j,k)})$............ (5)

$\mathrm{U}^i(\mathrm{c}) = \Sigma_{k=1}^{R} \mathrm{w}_k * (Q_k^i\ (\mathrm{c}) - \mathrm{Q}^i\mathrm{min(k)})/(\mathrm{Q}^i\mathrm{max(k)} - \mathrm{Q}^i\mathrm{min(k)})$.............. (6)

With $\mathrm{w}_k \in \mathrm{R}^+$ ($\Sigma_{k=1}^{R}\ (\mathrm{w}_k = 1)$) represent the weights (importance) of Q^i (or Q_k).

We took fair weights for all criteria.

Q_k^i (c): represents the aggregate value of the *kith* QOS criterion of composition *c*.

Q_k (s): represents the value of the *kith* QOS criterion of a concrete service *s*.

A composition is optimal if:

• It is feasible, i.e., it checks all global constraints

• it has the maximum value of the function U^i.

More formally, the user seeks to instantiate the abstract workflow of size *n* with a concrete composition c = $\{s_1^i, \ldots, s_n^i\}$ by choosing a concrete service $s_j^i \in \mathrm{S}_j$

such that:

1. $\mathrm{U}^i(\mathrm{c})$ is maximized
2. $Q_k^i\ (c) \geq \mathrm{Cons\ (k)}, \forall\ \mathrm{Cons\ (k)} \in \mathrm{CONS}$

Cons (k): represents a global constraint (a numerical constant) associated with the *K* criterion.

To integrate the global constraints into the objective function, we introduce a penalty function denoted *P*(*c*).

The objective of *P*(c) is to decrease the score of $\mathrm{U}^i(\mathrm{c})$ when one or several global constraints are violated.

$P(\mathrm{c}) = -\sum_{k=1}^{R} (\mathrm{D}_k)^2(\mathrm{c})$ .... (7)

$$\mathrm{D}_k\ (\mathrm{c}) = 0 \text{ if } \begin{cases} \mathrm{Q}_k^i(\mathrm{c}) \ \geq \ \mathrm{Cons(k)} \\ \left|\mathrm{Q}_k^i(c) - Cons(k)\right| \ otherwise \end{cases}$$

From this function, we can create a new function *f* that incorporates the global constraints and the QoS criteria at the same time:

$f(\mathrm{x}) = \ \mathrm{U}^i(x) + P(x)$.. (8)

Our goal is to find the solution *X* that maximizes the objective function *f* (also called fitness). However, finding the optimal solution requires enumerating all possible combinations of the candidate services. As mentioned in the problem statement, this problem is NP-hard. If a user's query concerns m classes of services (*m* tasks) and each class contains *l* candidate services, there are l^m possible combinations to consider. For this purpose, we propose the following optimization algorithm

3.6 Genetic Algorithm

Genetic Algorithms (GAs) are meta-heuristic search or optimization methods [28]. These techniques were originally inspired by the Darwinian principle of evolution by (genetic) selection.

A GA is based on a very abstract form of evolutionary processes to provide solutions to complex problems. Each GA operates on a population of artificial chromosomes. Each chromosome signifies a solution to the problem at hand and has a physical form. A physical form of the chromosome is an actual numerical measure that represents its performance as a solution to the specific problem.

The GA method starts with a randomly generated population of chromosomes. It then performs a selection and recombination process based on the shape of each chromosome. The parental genetic material is recombined to generate child chromosomes, producing a new generation. This process is iterated until a stopping criterion is reached. In this way, a GA develops the best solution to a given problem.

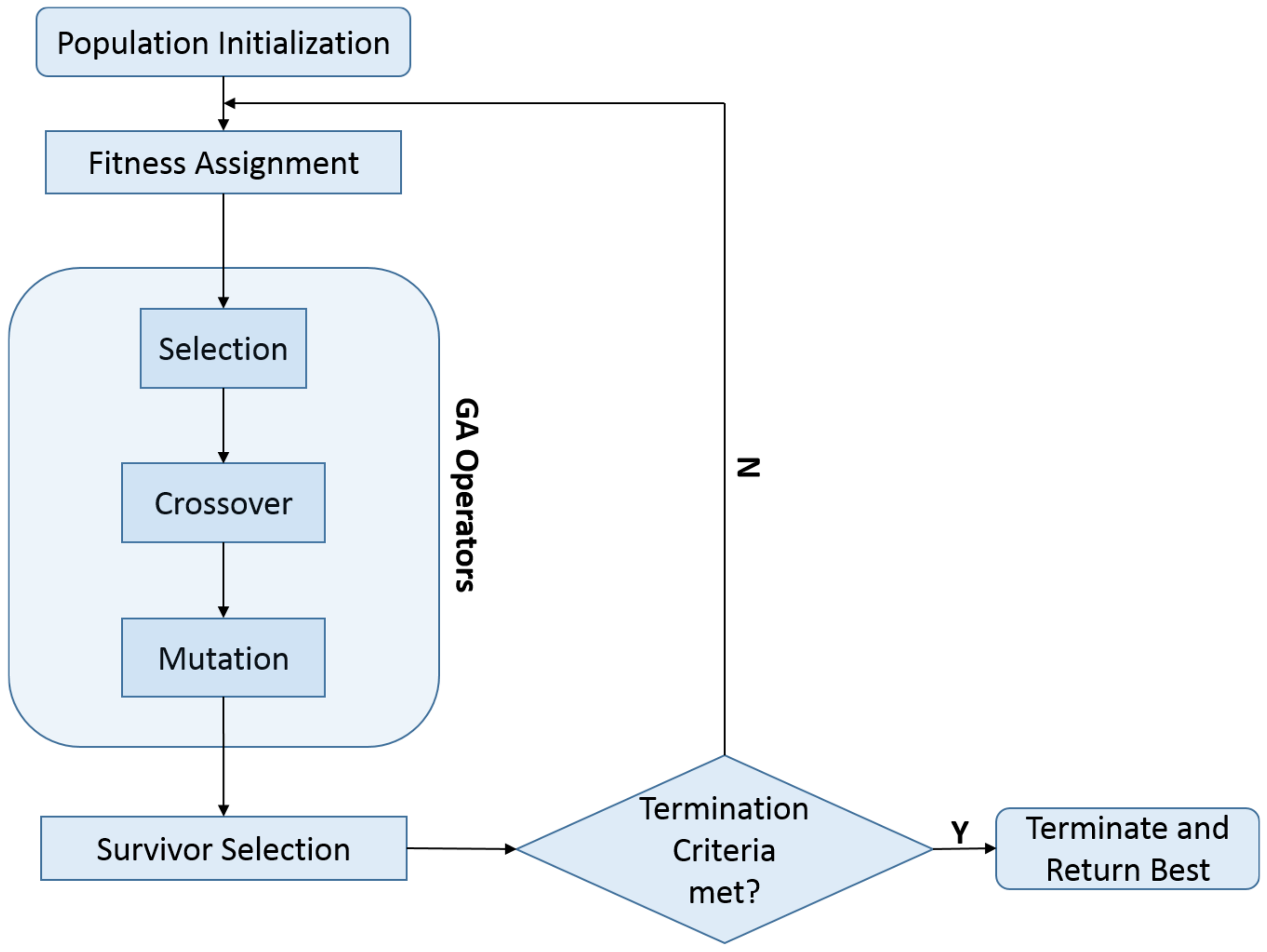


Figure 9: Genetic Algorithm Flowchart [31]

The genetic algorithm presented in figure 2 starts with a basic population which is usually composed of strings of characters each corresponding to a chromosome. The content of this initial population is generated randomly, each solution is given a score corresponding to its adaptation to the problem, and then a selection is made within this population. When two chromosomes have been selected, a cross is made with a crossing operator chosen according to a probability Pc. mutations are then made on a small proportion of individuals with a mutation probability Pm. This process provides us with a new population. The individuals resulting from these genetic operators will be inserted by an insertion method into the new population and a stop test will be carried out to check the quality of the individuals obtained. If this test is verified then the algorithm stops with an optimal solution otherwise the process is repeated a large number of times so as to initiate the principle of evolution.

3.7 Service selection using genetic algorithm

The first proposed algorithm is based on genetic algorithms [28], its reproduction phase is carried out with the binary tournament model.

The pseudo code is given as follows:

Algorithm 6: Service selection using GA

Entries:

The query R= <n=size-comp, BorneQOS1, BorneQOS2, BorneQOS3, BorneQOS4,

BorneQOS5> with:

✓ size-comp: represents the number of abstract tasks required by the user.// it is

also the size of the position of chromosome i: X_i (t).

✓ TerminalQOS1... TerminalQOS5: represent the global requirements of the user

imposed on the aggregated QOS values.

A service base B segmented into n classes (each service is characterized by R

QOS values).

Outings:

The best chromosome: MC // (a vector maximizing the function f)

```
1. Initialize the population of chromosomes, $P$, such that the position $X_i$(t) of each individual
   $p_i$ is random (with t = 0), chromosome number=|$P$|.
2. As long as ($t$<=$Tmax$) or (no convergence)
          Do
a. Evaluate the performance $f(X_i(t))$ of each chromosome P.
b. $P'$= reproduction($P$)// according to the binary tournament model
c. $P'$ = crossover ($P'$, crossover rate)// population change
d. $P$=mutation ($P'$, mutation rate)// avoidance of local minima
```

```
e.  MC= best (P, MC) // update of the best MC chromosome
        End_Do
3.  return MC
```

The "convergence" condition means that the average performance (in the sense of *f*) of two consecutive populations *Pt* and *Pt+1* is almost the same.

3.8 Conclusion

In this chapter, we have presented the theme of our project. A metaheuristic optimization approach for Web services selection and composition, where we also expressed the operation of our approach Genetic Algorithm.

The next chapter will be devoted to the implementation of this algorithms and the description of the results obtained.

Chapter 4: Implementation

4.1 Introduction

In the previous chapter we presented the study and design of the metaheuristics and genetic algorithm (GA). After having presented in details our approach to select Web services taking into account the quality-of-service criteria, this chapter will be dedicated to the implementation phase.

The implementation of any program or application requires a set of steps, which plays a very important role in its performance. One of the most important steps is the working environment, the choice of the latter must be by awareness, according to the structures necessary for the implementation. As well as the working tools that facilitate the task of programming and using the data.

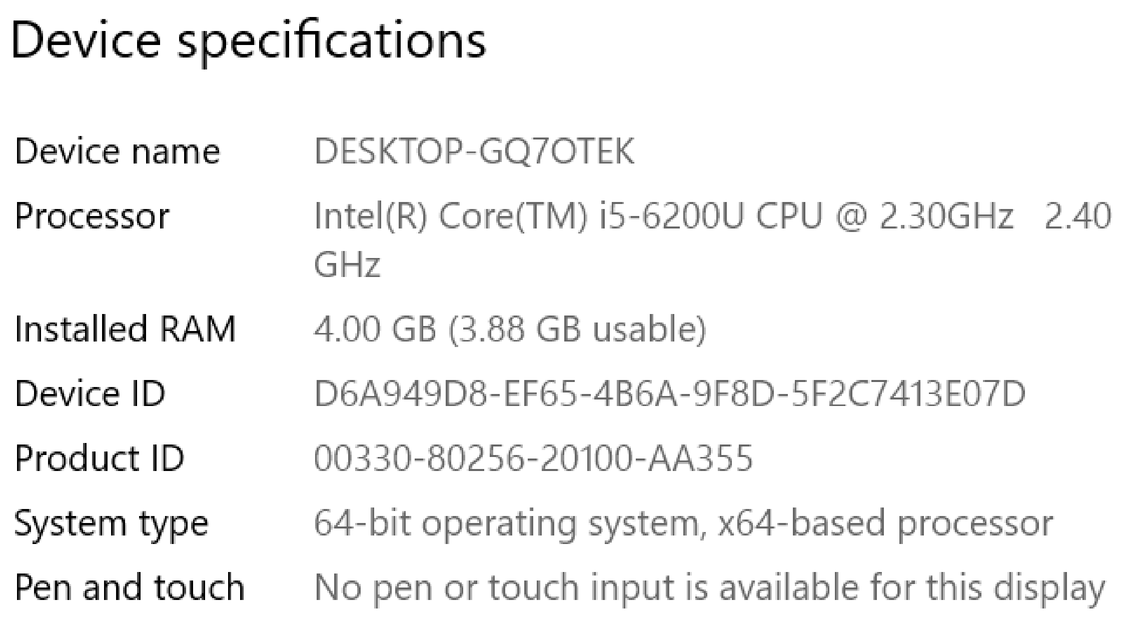
We will show how we have realized and implemented our system. We start with the presentation of the software environment used, through the presentation of the tools and the programming language.

4.2 Development environment and tools

4.2.1 Development environment

Before starting the implementation of our application, we will first specify the programming languages and tools used that we thought were a good choice given the advantages they offer.

To realize our system, we used the following computer i5 Windows 10 64 bits with the following specifications:


Device specifications

Device name	DESKTOP-GQ7OTEK
Processor	Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz
Installed RAM	4.00 GB (3.88 GB usable)
Device ID	D6A949D8-EF65-4B6A-9F8D-5F2C7413E07D
Product ID	00330-80256-20100-AA355
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 4.1: Software environment used

4.2.1.1 PyCharm IDE

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment specifically for the Python programming language, web, and data science development... It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also an educational version, as well as a Professional Edition with extra features (released under a subscription-funded proprietary license).

Features

- Intelligent Code Editor:

PyCharm's smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages and more. Take advantage of language-aware code completion, error detection, and on-the-fly code fixes!

- Smart Code Navigation:

  Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.

- Fast and Safe Refactorings:

  Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable or Method, and other refactorings. Language and framework-specific refactorings help you perform project-wide changes.

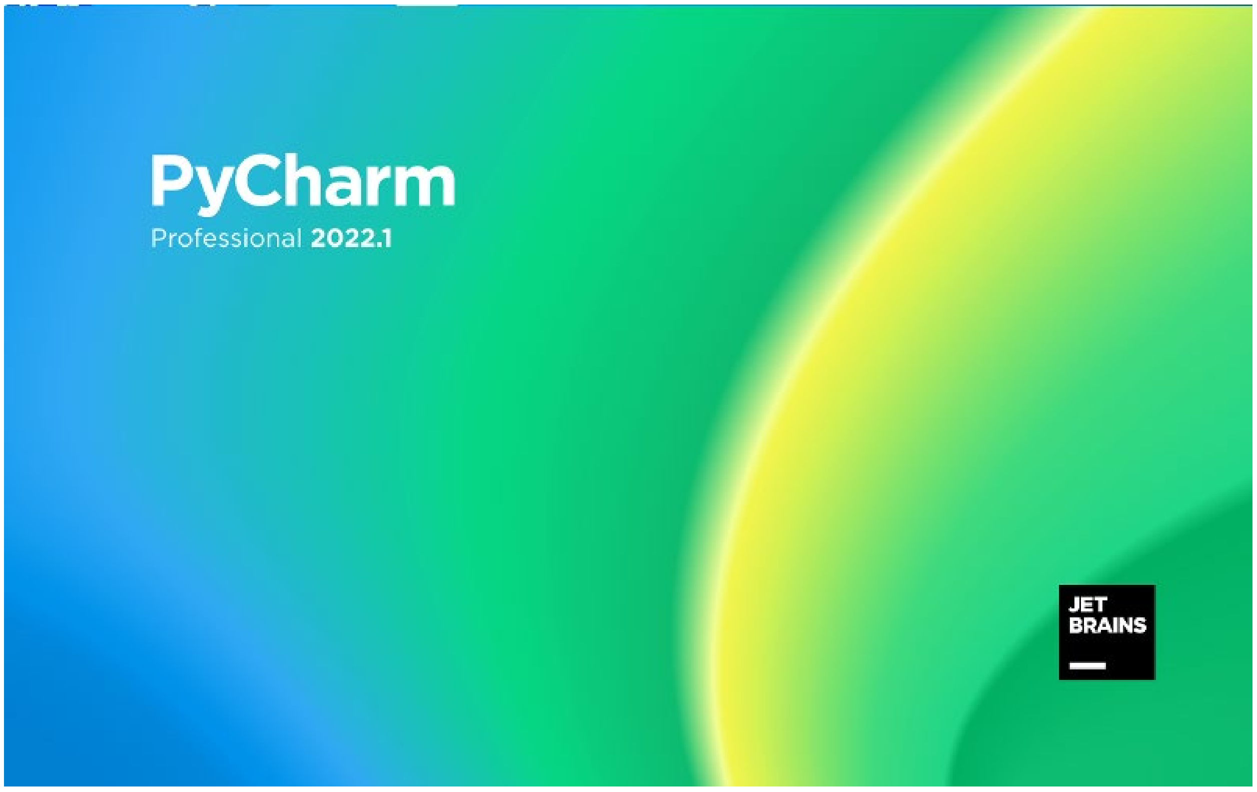
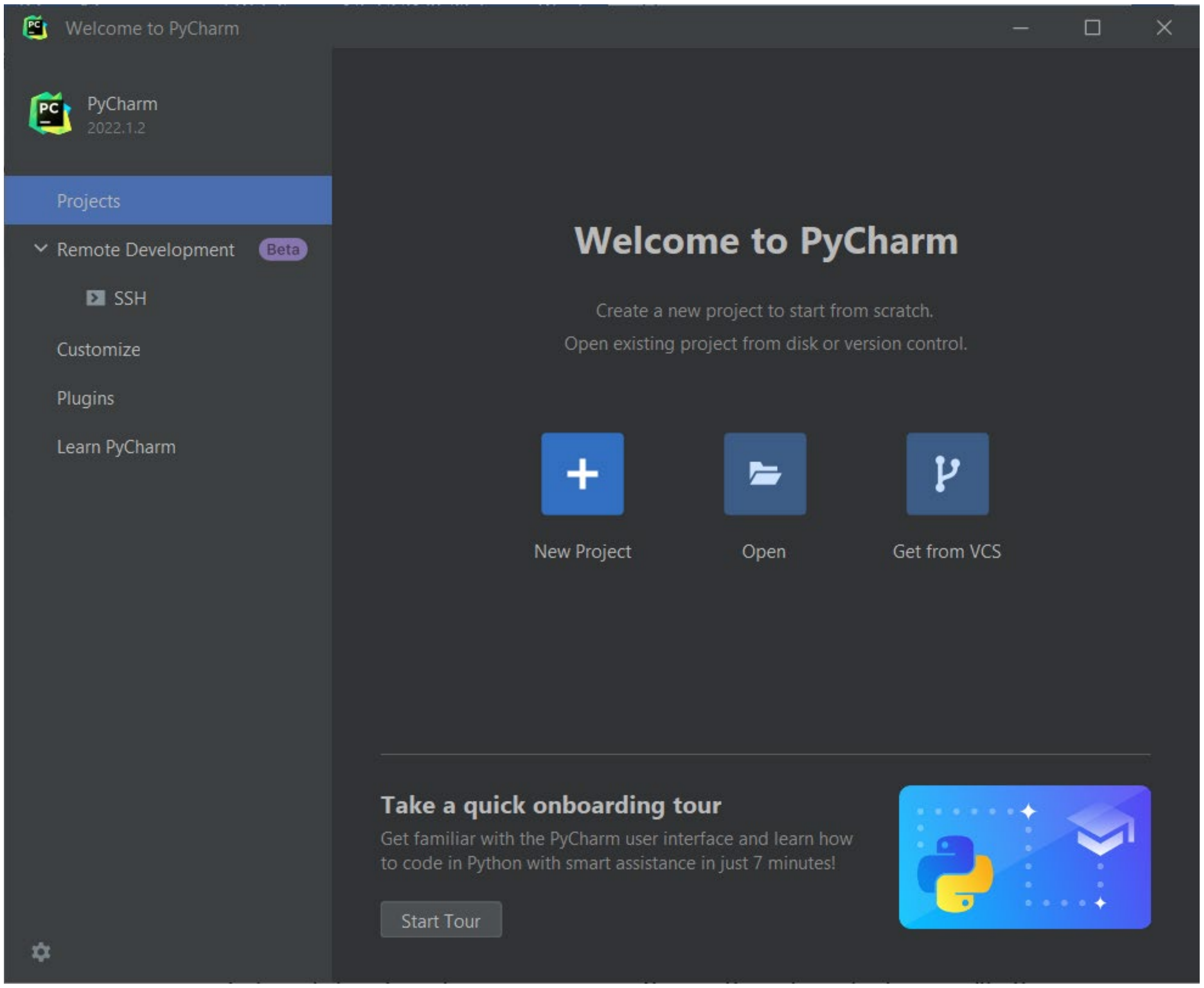


Figure 10: PyCharm IDE



Welcome to PyCharm

Create a new project to start from scratch.
Open existing project from disk or version control.

Take a quick onboarding tour

Get familiar with the PyCharm user interface and learn how to code in Python with smart assistance in just 7 minutes!

Figure 11: PyCharm Main Interface

4.2.1.2 Python Language

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation [32].

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library [32].

Figure 12: Python Logo and Symbol

Easy To Learn and Readable Language

Python is extremely easy to learn. Its syntax is super simple and the learning curve of Python is very smooth. It is extremely easy to learn and code in Python and the indentation used instead of curly braces in Python makes it very easy to read Python code. Perhaps, because of this a lot of schools and universities, and colleges are teaching Python to their students who are beginning their journey with coding [33].

Interpreted Language

Python is an interpreted language (an interpreted language is a programming language that is generally interpreted, without compiling a program into machine instructions. It is one where the instructions are not directly executed by the target machine, but instead, read and executed by some other program known as the interpreter) and an IDLE (Interactive Development Environment) is packaged along with Python. It is nothing but an interpreter which follows the REPL (Read Evaluate Print Loop) structure just like in Node.js. IDLE executes and displays the output of one line of Python code at a time. Hence, it displays errors when we are running a line of Python code and displays the entire stack trace for the error [33].

Dynamically Typed Language

Python is a dynamically typed language. In other words, in Python, we do not need to declare the data types of the variables which we define. It is the job of the Python interpreter to determine the data types of the variables at runtime based on the types of the parts of the expression. Though it makes coding easier for programmers, this property might create runtime errors. To be specific, Python follows duck typing. It means that "If it looks like a duck, swims like a duck and quacks like a duck, it must be a duck." [33].

Open Source and Free

Python is an open-source programming language and one can download it for free from Python's official website. The community of Python users is constantly contributing to the code of Python in order to improve it [33].

Large Standard Library

One of the very important features because of which Python is so famous in today's times is the huge standard library it offers to its users. The standard library of Python is extremely large with a diverse set of packages and modules like itertools, functools, operator, and many more with common and important functionalities in them. If the code of some functionality is already present in these modules and packages, the developers do not need to rewrite them from scratch, saving both time and effort on the developer's end. Moreover, the developers can now focus on more important things concerning their projects. Also, Python provides the PyPI (Python Package Index) which contains more packages that we can install and use if we want even more functionality [33].

High-Level Language

A high-level language (HLL) is a programming language that enables a programmer to write programs that are more or less independent of a particular type of computer. These languages are said to be high level since they are very close to human languages and far away from machine languages. Unlike C, Python is a high-level language. We can easily understand

Python and it is closer to the user than middle-level languages like C. In Python, we do not need to remember system architecture or manage the memory [33].

Object Oriented Programming Language

Python supports various programming paradigms like structured programming, functional programming, and object-oriented programming. However, the most important fact is that the Object-Oriented approach of Python allows its users to implement the concepts of Encapsulation, Inheritance, Polymorphism, etc. which is extremely important for the coding done in most Software Industries as objects map to entities in the real world and a lot of real-world problems can be solved using the Object-Oriented Approach [33].

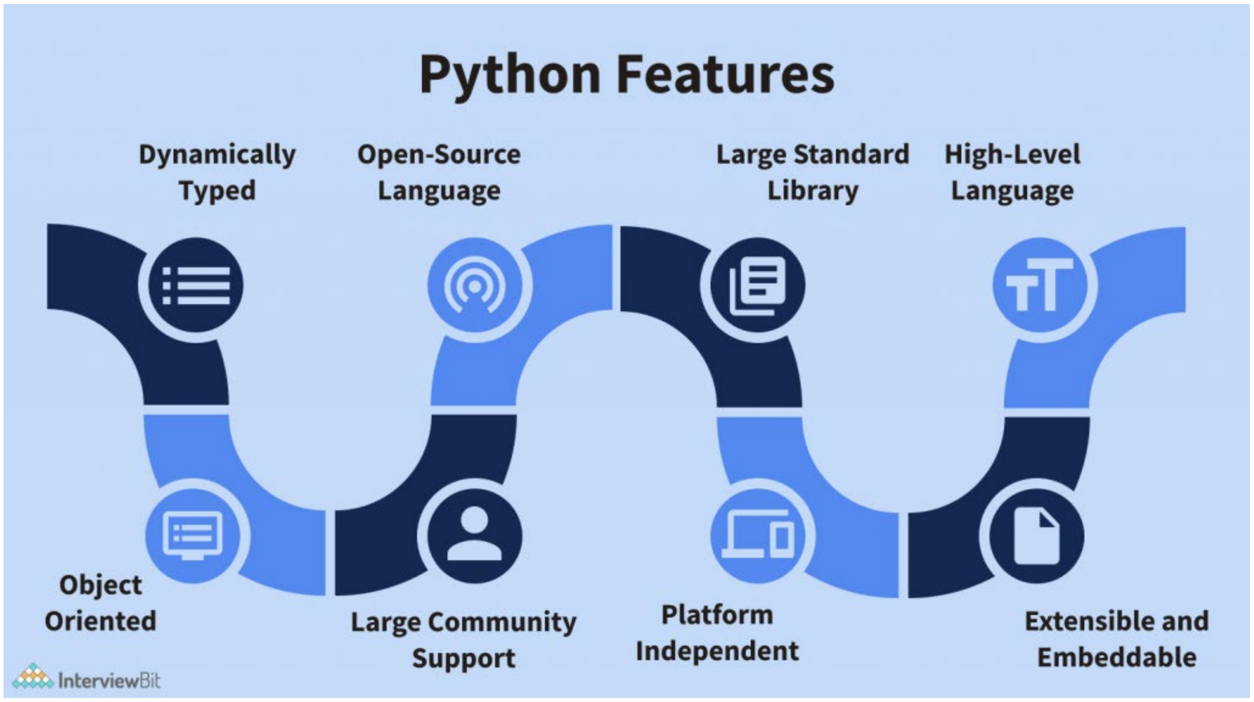


Figure 13: Python Features [33]

4.2.1.3 Python Libraries

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer. As we don’t need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc [34].

Python standard library

The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most of the Python Libraries are written in the C programming language. The Python standard library consists of more than 200 core modules. All these works together to make Python a high-level programming language. Python Standard Library plays a very important role. Without it, the programmers can't have access to the functionalities of Python. But other than this, there are several other libraries in Python that make a programmer's life easier. Let's have a look at some of the commonly used (including some used in our implementation):

Numpy: The name "Numpy" stands for "Numerical Python". It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.

TensorFlow: This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.

Matplotlib: This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

Pandas: Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like

Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

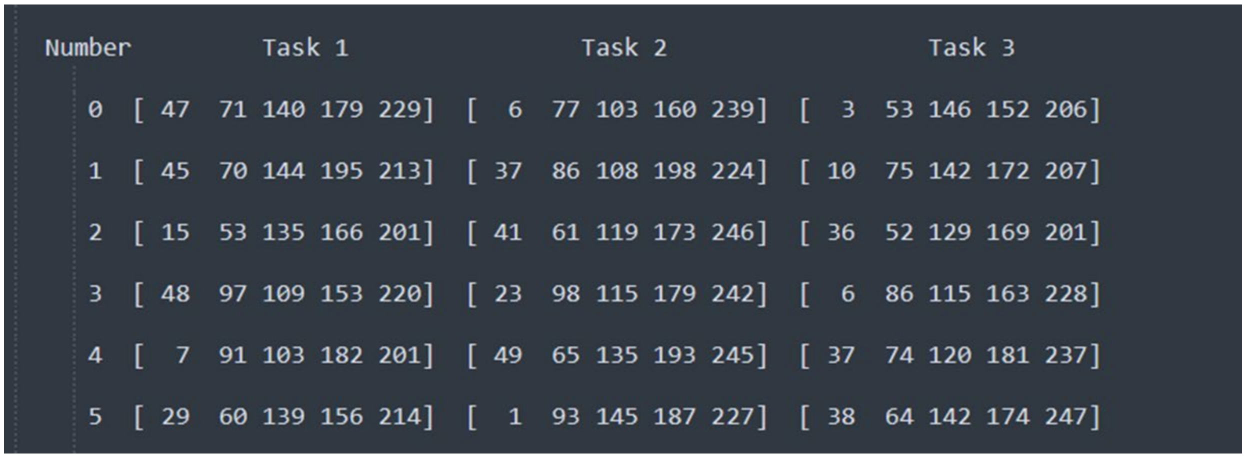
SciPy: The name "SciPy" stands for "Scientific Python". It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

Scrapy: It is an open-source library that is used for extracting data from websites. It provides very fast web crawling and high-level screen scraping. It can also be used for data mining and automated testing of data.

Scikit-learn: It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

4.3 Presentation of the implementation

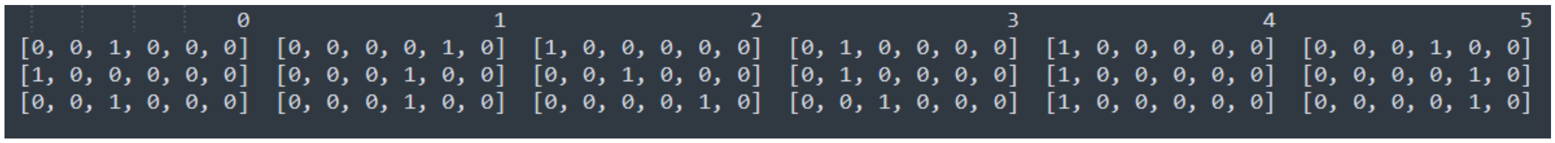
4.3.1 Generating Data



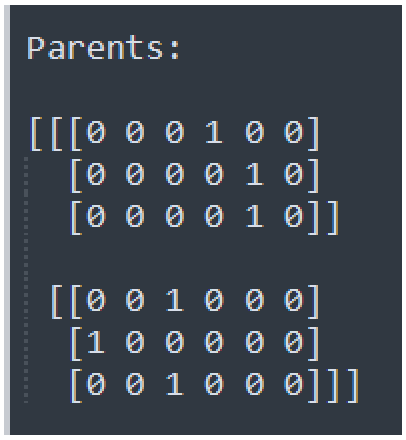
```
Number        Task 1                  Task 2                  Task 3

   0  [ 47  71 140 179 229]  [  6  77 103 160 239]  [  3  53 146 152 206]
   1  [ 45  70 144 195 213]  [ 37  86 108 198 224]  [ 10  75 142 172 207]
   2  [ 15  53 135 166 201]  [ 41  61 119 173 246]  [ 36  52 129 169 201]
   3  [ 48  97 109 153 220]  [ 23  98 115 179 242]  [  6  86 115 163 228]
   4  [  7  91 103 182 201]  [ 49  65 135 193 245]  [ 37  74 120 181 237]
   5  [ 29  60 139 156 214]  [  1  93 145 187 227]  [ 38  64 142 174 247]
```

4.3.2 Generating Population of Bits:

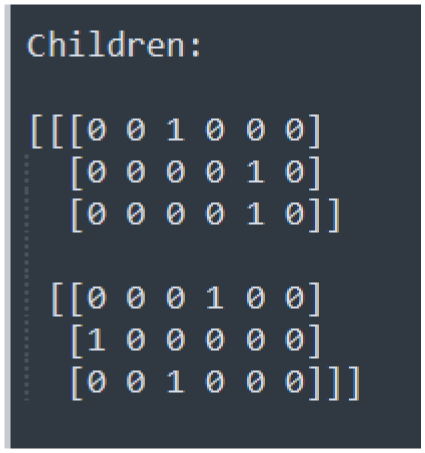
Printing Population:


```
                   0                         1                         2                         3                         4                         5
[0, 0, 1, 0, 0, 0]  [0, 0, 0, 0, 1, 0]  [1, 0, 0, 0, 0, 0]  [0, 1, 0, 0, 0, 0]  [1, 0, 0, 0, 0, 0]  [0, 0, 0, 1, 0, 0]
[1, 0, 0, 0, 0, 0]  [0, 0, 0, 1, 0, 0]  [0, 0, 1, 0, 0, 0]  [0, 1, 0, 0, 0, 0]  [1, 0, 0, 0, 0, 0]  [0, 0, 0, 0, 1, 0]
[0, 0, 1, 0, 0, 0]  [0, 0, 0, 1, 0, 0]  [0, 0, 0, 0, 1, 0]  [0, 0, 1, 0, 0, 0]  [1, 0, 0, 0, 0, 0]  [0, 0, 0, 0, 1, 0]
```

Selecting Parents:

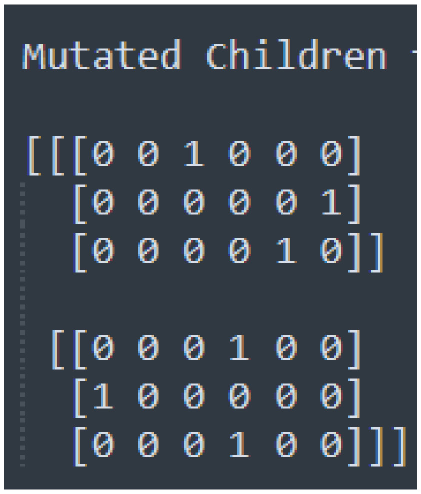

```
Parents:

[[[0 0 0 1 0 0]
  [0 0 0 0 1 0]
  [0 0 0 0 1 0]]

 [[0 0 1 0 0 0]
  [1 0 0 0 0 0]
  [0 0 1 0 0 0]]]
```

Creating Children:



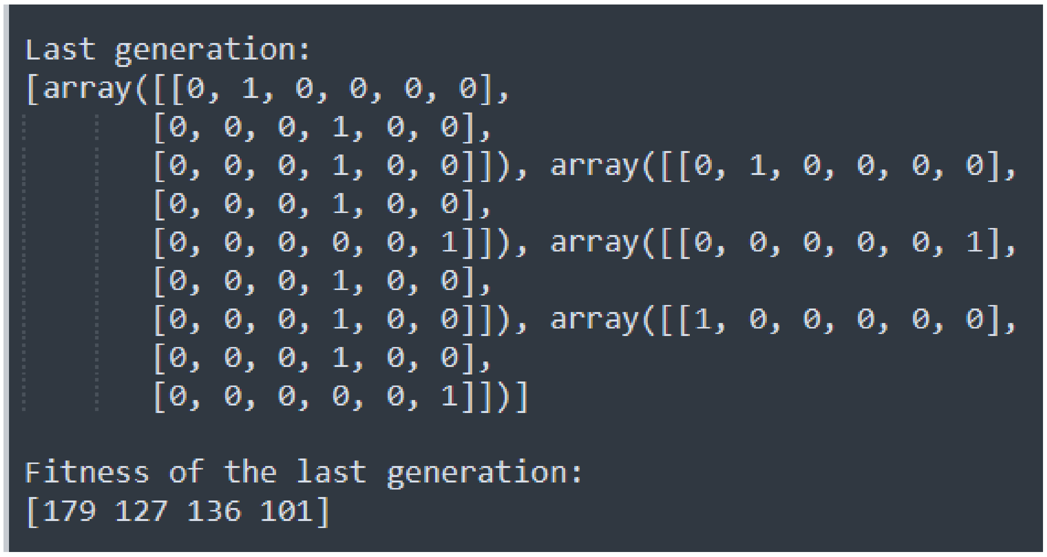
```
Children:

[[[0 0 1 0 0 0]
  [0 0 0 0 1 0]
  [0 0 0 0 1 0]]

 [[0 0 0 1 0 0]
  [1 0 0 0 0 0]
  [0 0 1 0 0 0]]]
```

Mutating Children:

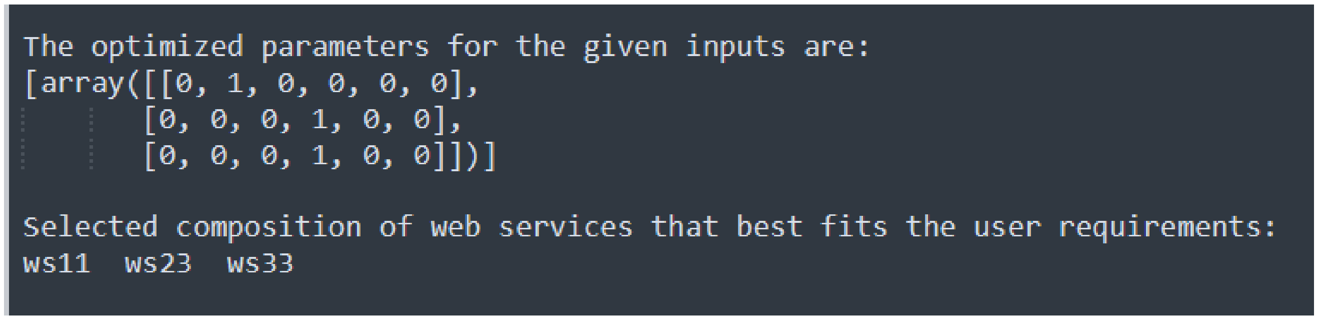


```
Mutated Children

[[[0 0 1 0 0 0]
  [0 0 0 0 0 1]
  [0 0 0 0 1 0]]

 [[0 0 0 1 0 0]
  [1 0 0 0 0 0]
  [0 0 0 1 0 0]]]
```

The last generation of the population and selection of the best composition of web services:



```
Last generation:
[array([[0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0]]), array([[0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1]]), array([[0, 0, 0, 0, 0, 1],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0]]), array([[1, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1]])]

Fitness of the last generation:
[179 127 136 101]
```



```
The optimized parameters for the given inputs are:
[array([[0, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0]])]

Selected composition of web services that best fits the user requirements:
ws11  ws23  ws33
```

4.4 Conclusion

We have presented some details regarding the realization of our code, choosing the python programming language for its simplicity and portability.

General conclusion

The web service selection is a primordial step in the support of dynamic compositions. Web service composition is used to compose the existing web services to get a new value-added function, which has become one of the most features of web services, because it creates new functionalities by aggregating different services based on specific workflow. With the increasing number of web services providing similar functionalities, the QoS is becoming an important criterion of the selection of the best available services.

In our work, we have presented the web services selection problem as combinatorial optimization problem where the metaheuristic methods are necessary to tackle the challenge of QoS-aware web service selection. For this context, we have explored the potential of meta-heuristics for the web service selection problem. We have adapted the concepts of genetic algorithms to propose a web service selection algorithm based-GA, in web service composition. The proposed algorithm has been implemented and evaluated by some prepared examples. The evaluation results were satisfactory

However, this research could be improved in certain points which we consider as prospects

- Using web service's dataset for validation, and testing

- Adapt the concept of Pareto multi-objective optimization to take into account several criteria of QoS simultaneously.

Références

[1] P. Siarry, Metaheuristics.

[2] Q. L. X. B. A. e. a. Yu, «Deploying and managing Web services: issues, solutions, and directions. The VLDB Journal 17,» *The International Journal on Very Large Data Bases,* p. 537–572, 2008.

[3] R. Abrahiem, «A new generation of middleware solutions for a near-real-time data warehousing architecture,» *2007 IEEE International Conference on Electro/Information Technology,* pp. 192-197, 2007.

[4] A.-P. J. Eldorina-Andreea Alergus, «Web Service Composition,» 19 10 2009. [En ligne]. Available: https://www.slideshare.net/eldorina/web-services-composition-2282938. [Accès le 24 05 2022].

[5] «Web Services Architecture,» World Wide Web Consortiom, 11 02 2004. [En ligne]. Available: http\://www.w3c.com/TR/ws-arch/. [Accès le 01 06 2022].

[6] «Web Services - IBM,» IBM Corporation, [En ligne]. Available: https://www.ibm.com/docs/en/was/9.0.5?topic=services-web. [Accès le 01 06 2022].

[7] T. W. S. I. Company, «Introduction to Web Services,» Systinet Corporation, Cambridge, 2002.

[8] Y. H. Khadidja Salah Mansour, «ATL Based Refinement of WS-CDL Choreography into BPEL Processes,» *Modelling and Implementation of Complex Systems,* vol. 64, pp. 329-343, 2019.

[9] I. S. G. Heather Kreger, «Web Services Conceptual Architecture,» May 2001. [En ligne]. Available: https://www.csd.uoc.gr/~hy565/docs/pdfs/papers/wsca.pdf. [Accès le 12 06 2022].

[10] Javatpoint, «Restful Web Services Architecture of Web Services,» [En ligne]. Available: https://www.javatpoint.com/restful-web-services-architecture-of-web-services. [Accès le 18 06 2022].

[11] G. M. W. a. R. B. Ahmad, «Security protection using simple object access protocol (SOAP) messages techniques,» *2008 International Conference on Electronic Design,* pp. 1-6, 2008.

[12] M. S. a. Y. K. F. Balde, «Performance analysis of Web services in mobile environment using SWN and WAP protocol,» *First International Conference On Parallel, Distributed and Grid Computing (PDGC),* pp. 290-295, 2010.

[13] [En ligne]. Available: http://hedayatblog.blogspot.com/2010/10/course-work-1-activity-21.html. [Accès le 10 06 2022].

[14] «SAMS Teach Yourself Web Services in 24 Hours,» [En ligne]. Available: http://bedford-computing.co.uk/learning/wp-content/uploads/2016/08/SAMS-Teach-Yourself-Web-Services-in-24-Hours.pdf.

[15] «SOAP Explanation,» 2010. [En ligne]. Available: https://www.dotnetcrack.com/post/2010/10/07/SOAP-Explanation.aspx. [Accès le 06 2022].

[16] X. Liu, «Integrating Protein Data Resources through Semantic Web Services,» 2006.

[17] M. R. Gayathridevi M, «Semantic web services — a survey.,» *Int J Sci Eng Res,* 2013 .

[18] L. D. J. M. E. P. T. H. F. Cabral, «Approaches to Semantic Web Services: an Overview and Comparisons.,» *European semantic web symposium,* pp. 225-239, 2004.

[19] D. P. M. M. S. B. M. M. D. M. D. P. B. P. T. R. S. M. S. M. e. a. Martin, «Bringing semantics to web services : The owl-s approach.,» 2004.

[20] M. A. C. D. Y. M. Hadjila Fethallah, «QoS-aware Service Selection Based on Genetic Algorithm».

[21] A. ,. L. .. Tang Maolin, « A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition,» *IEEE Computer Society,* 2010.

[22] A. A. a. A. Touir, «WEB SERVICE COMPOSITION PROCESSES: A COMPARATIVE STUDY,» *International Journal on Web Service Computing (IJWSC),* vol. 7, n° %11, March March 2016.

[23] N. A. H. A. M. B. M. Fariss, «An Improved Approach for QoS Based Web Services Selection Using Clustering,» *Advances in Science, Technology and Engineering Systems Journal,* vol. 6, n° %12, pp. 616-621, 2021.

[24] S. Khaddaj, «Cloud Computing: Service Provisioning and User Requirements,» *2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science,* pp. 191-195, 2012.

[25] A. E. S. a. A. S. Bassam AL-Shargabi, « Web Service Composition Survey: State of the Art Review».

[26] C. Zeginis, «Monitoring the QoS of Web Services using SLAs - Computing metrics for composed services».

[27] H. M. O. M. a. A. E.-B. Walaa Nagy, «A Flexible Tool for Web Service Selection in Service Oriented Architecture,» *International Journal of Advanced Computer Science and Applications(IJACSA),* 2011.

[28] J. H. Holland, «Outline for a Logical Theory of Adaptive Systems,» *The Journal of the ACM (JACM),* vol. 9, n° %13, pp. 297-314, 1962.

[29] R. L. X. Z. Xinchao Zhao, «Advances on QoS-aware web service selection and composition with nature-inspired computing,» *IET The Institution of Engineering and Technology,* 2019.

[30] M. R. T. a. N. W. Alrifai, «A hybrid approach for efficient web service composition with end-to-end qos constraints,» *ACM Transactions on the Web (TWEB),* 2012.

[31] Anand Deshpande and Manish Kumar. 2018. Artificial Intelligence for Big Data: Complete guide to automating Big Data solutions using Artificial Intelligence techniques. Packt Publishing..

[32] P. S. Foundation, About Python, 2012.

[33] «Top 10 Features of Python You Must Know,» 21 06 2022. [En ligne]. Available: https://www.interviewbit.com/blog/features-of-python/. [Accès le 22 06 2022].

[34] GeeksforGeeks, «Libraries in Python,» 21 10 2021. [En ligne]. Available: https://www.geeksforgeeks.org/libraries-in-python/. [Accès le 22 06 2022].

[35] S. A. Morteza Khani Dehnoi, «Automatic QoS-aware Web Services Composition,» *Int. J. Nonlinear Anal. Appl. 12 (2021) No. 1, 87-109,* 2021.

[36] A. D. K. Manisha A. Kumbhar1, «Semantic Web for E-Governance,» *Journal of Algorithms & Computational Technology,* vol. 4, n° %14, p. 533, 2009.

[37] G. Spieler, «What is the Web,» Medium, 03 05 2021. [En ligne]. Available: https://gspieler.medium.com/what-is-the-web-6c7177a41efa. [Accès le 24 05 2022].

[38] MDN, «World Wide Web,» MDN, 27 04 2022. [En ligne]. Available: https://developer.mozilla.org/en-US/docs/Glossary/World_Wide_Web. [Accès le 24 05 2022].

[39] A. Kuchling, «PEP 206 - Python Advanced Library,» Python.org, 14 07 2000. [En ligne]. Available: https://peps.python.org/pep-0206/. [Accès le 22 06 2022].

[40] D. Kuhlman, A Python Book: Beginning Python, Advanced Python, and Python Exercises. Section 1.1., 2012.