Received July 6, 2021, accepted August 2, 2021, date of publication August 5, 2021, date of current version August 16, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3102668*

# Optimization of Business Process Execution in Services Architecture: A Systematic Literature Review

**TOMASZ GÓRSKI[1], (Member, IEEE), AND ADRIAN P. WOŹNIAK[2]**
[1]Department of Computer Science, Polish Naval Academy (PNA) of the Heroes of Westerplatte, 81-127 Gdynia, Poland
[2]Institute of Computer and Information Systems, Military University of Technology, 00-908 Warsaw, Poland

Corresponding author: Tomasz Górski (t.gorski@amw.gdynia.pl)

This work was supported by the Statutory Funds of the Department of Computer Science, Polish Naval Academy (PNA) of the Heroes of Westerplatte, through the Project Architectural views model of cooperating information technology (IT) systems.

**ABSTRACT** Web services have become a standard way to provide functions of information systems. The number of web services grows rapidly with the increasing popularity of microservices architecture. In consequence, many business processes are executed entirely through web services. Therefore, optimizing the performance of business process execution may bring many benefits. There are many optimization methods in this area. Our systematic literature review aims to introduce available methods to researchers interested in the optimization of business process execution. We queried four databases: ACM, IEEE Xplore, Science Direct, and Springer. Out of 12150 initially found papers, we have selected 128 for the review. We have grouped methods presented in those papers into three stages of business process optimization: Resource Allocation, Service Composition, and Service Scheduling. Service Composition attracts the largest group of researchers with a vast majority of 119 articles in it. Moreover, the most popular are genetic algorithms. In general, researchers mainly propose heuristic methods that optimize business processes during run-time. We see the potential for further exploration at both Resource Allocation and Service Scheduling stages.

**INDEX TERMS** Service-oriented architecture, business process, optimization, reliability, micro-services.

## I. INTRODUCTION

Service Oriented Architecture (SOA) is an approach that views systems as loosely coupled components. Each component may be implemented in different technology and running on a different server. Components are responsible for the realization of certain business functions of the organization. Components can support business processes that cross many business areas of the organization. Components communicate with each other through services. A service is a function of a system that gives value to an organization. A component that realizes a service is called the provider. On the other hand, the component that uses the service is called the consumer. Each service is a concise function that gives business value to the consumer. Granularity is a feature that describes the size of the scope for which a service is responsible. Services that have too much responsibility are not usable. Services with too little liability are reusable but costly to maintain. It is best practice to design services whose scope corresponds to the step in the business process of the organization. Autonomy is the ability to provide value without relying on external resources. In order to achieve that the component should have a well-defined, separate area of responsibility. Both, service providers and consumers are components that are independent and loosely coupled. It means that each component is software that can be developed separately and is technologically independent. Components cooperate with each other through a service broker which is usually Enterprise Service Bus (ESB). ESB is software that also mitigates technological differences among components. This approach allows on use of existing software in companies, adds a service interface layer, and integrates it with new software to give new business value. Service reusability is one of the essential paradigms of SOA. It is beneficial for the organization when the service can be used in different business processes. In a service architecture that supports the implementation of an organization's

The associate editor coordinating the review of this manuscript and approving it for publication was Taehong Kim.

business processes, an element that manages these processes is indispensable. The solution is a Business Process Management (BPM) engine, that may control the sequence of services invocations. That process is called orchestration. It means that the BPM engine conducts SOA components, the members of the orchestra. Most suppliers of enterprise service bus software deliver their products with BPM functionality within. The service-oriented approach developed rapidly within the last years. Components have become smaller and lighter which transforms the approach into microservices. In the microservices approach, an enterprise bus is usually replaced with a light service broker, whose only purpose is to resend service invocations to correct providers. Regardless of how service architecture is implemented, a service should provide added business value and be usable in a business process. In the paper, we assume that the business process is a sequence of service invocations organized through orchestration or choreography. The growing complexity of architecture comes in pair with additional problems. One of such problems is the optimization of the performance of business processes realized in the service architecture approach. It is not enough for software components to work well separately. They should work effectively together as a system that supports the business processes of a company. Service-Oriented Architecture is a very popular approach, so it is not surprising that many publications are available in literature considering that subject.

Many literature reviews are concerned with business processes. Some of them relate to the modeling of business processes. For example, Amjad *et al.* [2] investigate the latest applications of Event-Driven Process Chain, a well-known business process modeling language, for the modeling and verification of business requirements. Some analyze deviations from the standard operating procedure of business processes. In that field, Omair and Alturki [3] conducted a systematic content analysis of the literature on fraud detection metrics in business processes. Recently, there is more and more work on blockchain technology and decentralized management of business processes. García-García *et al.* [4] reviewed literature to know what proposals exist to improve any stage of business process management using blockchain technology. The microservices architecture is more and more widely used and present in scientific works. Rasheedh and Saradha [5] present a review of micro-services architecture papers and paid special attention to agility and scalability. There are also literature reviews that focus on one type of algorithm. In one of the recent works, Katoch *et al.* [8] describe current advances and future challenges for genetic algorithms.

Our literature review concentrates on the optimization of business processes implemented in services architecture. We have taken into account both the Service-Oriented Architecture and microservices approaches. Moreover, we are interested in the optimization of service requests in the context of business process realization. We have not considered single, isolated service requests. In addition, we have included, in the review, only those papers from the literature that showed solutions for assessing and ensuring the reliability of the information technology system built in the service architecture. However, we have excluded articles on the assessment of the reliability of a single component. What is important, we have not concentrated on the single approach but aimed to identify available approaches to solving the problems in the field of optimization of business process execution.

The paper is structured as follows. Section 2 presents our research method. In section 3 we introduce notions of elements in service-oriented architecture and business processes. That section outlines three optimization stages of business process execution. Section 4 describes the elements of the genetic algorithm and the standard procedure used in it. The following section contains a detailed description of our literature review results. In section 6 we have presented the taxonomy of heuristic algorithms applied to the optimization of business process execution and a detailed analysis of genetic algorithms found in works encompassed by the review. Section 7 presents the discussion and limitations. In that section, we have also answered research questions. The last section shows conclusions.

## II. RESEARCH METHOD

The section presents steps that were undertaken according to the method presented by Kitchenham and Charters [1]. First, we formed research questions to be answered by our systematic review.

The paper reviews the literature on the subject of optimization of business processes supported by services. Therefore, we have proposed the following research questions:

- RQ1 - What methods of optimizing business processes in SOA have been published?
- RQ2 - What approaches to the fallibility of servers and components have been proposed?
- RQ3 - What kind of heuristic and deterministic algorithms have been implemented in those methods?
- RQ4 - What are the strengths and weaknesses of those methods?

We performed a systematic review of the literature which gave answers to the presented above research questions. Classification of literature has been proposed, which allowed performing statistical analysis of articles. In addition, every type of approach presented in an article or group of articles is presented. Secondly, inclusion and exclusion criteria were proposed to select papers of interest. Those criteria were subject to change during the process review. If the criteria were insufficient to decide whether to reject or accept an article, we added a new criterion to the list. Then data sources and search strategies were set and search questions to databases were proposed. Then, we performed a paper selection and quality assessment to double-check the results. Lastly, all selected articles were read and analyzed. The results of that analysis are the main content of the paper.

The scope of the study according to the PICO template is as follows:

- Population: SOA architecture, business processes in SOA, reliability of services,
- Intervention: Methods used to optimize the allocation of components to servers in SOA architecture that takes reliability into account,
- Comparison: Different methods of allocation optimization, especially in areas of used algorithms and information taken into account,
- Outcome: Complete spectrum of methods used to optimize the allocation of components to servers in SOA.

The main context of our study is the optimization of business processes that are supported by services architecture. Each service can be executed with different efficiency, according to the selected method of optimization of business processes.

### A. INCLUSION AND EXCLUSION CRITERIA

We proposed inclusion and exclusion criteria to ensure that the reasoning behind decisions of inclusion to the review is clear. Those criteria were subject to refinement during the selection of papers. We were not able to predict in advance, in inclusion or exclusion criteria, all aspects that appeared in articles. We have included publications that possess the following qualities (inclusion criteria):

- IN1 - relate to the optimization of business processes implemented in service-oriented architecture,
- IN2 - describe an environment that allows simulating business processes executed by services,
- IN3 - demonstrate a way of reliability evaluation of service-oriented system (papers about the evaluation of single component reliability were not included),
- IN4 - relate to the optimization of service requests in the context of business process realization (papers about single request optimization were excluded).

Papers that are not directly about optimization of business process realization but can optimize those processes (e.g.: composite services, service orchestration) have been also included.

We have excluded publications that possess the following qualities (exclusion criteria):

- EX1 - focus on infrastructure network layer (HTTP protocols and routers and proxy servers topology),
- EX2 - focus on problems of mobile environments in the context of service composition,
- EX3 - relate to the optimization of the development process of SOA systems,
- EX4 - concentrate on Business Process Model and Notation (BPMN) and its usage in SOA solutions,
- EX5 - treat about the deployment process of components in SOA,
- EX6 - propose optimization while reducing the quality of service (e.g. faster responses, but with less precision),
- EX7 - describe simulators of SOA systems that assume that resources are infinite,
- EX8 - describe business process optimization that changes the business process,
- EX9 - show optimization of energy consumption,
- EX10 - deal with service discovery optimization,
- EX11 - describe the optimization of embedded systems services utilization.

In general, we have excluded papers about optimization, simulation, or reliability evaluation of a single component. We have focused on papers about the SOA system as a whole (set of components and ESB).

### B. DATA SOURCES AND SEARCH STRATEGY

The next step was a selection of databases to be searched. To select databases, we analyzed several systematic reviews that concern IT system architecture. As result, we selected the four most popular databases: ACM, IEEE Xplore, Science Direct, and Springer.

In order to find relevant papers, we have formulated a search clause. At first, the search condition was defined as follows:

*((allocation AND (components OR modules OR resource)) OR (scheduling AND (task OR service OR "business process"))) AND (SOA OR ESB) AND (optimization OR optimization)*

However, during the in-depth analysis of results, it evolved into a more elaborated clause that gave more relevant results. In the end, the search condition took the following form:

*(SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (optimisation OR optimization OR optimize OR allocate OR allocation OR scheduling OR schedule OR "business process" OR reliability OR dependability OR reliable OR dependable OR simulation OR simulate OR genetic)*

IEEE Xplore and Science Direct have limitations of the number of words used in search condition, so we divided it into two parts:

- *SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (optimisation OR optimization OR optimize OR allocate OR allocation OR scheduling OR schedule OR "business process"*,
- *SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (dependability OR reliable OR dependable OR simulation OR simulate OR genetic*.

In 2021, Science Direct had limited to 8 the number of logical operators. So, the search condition for Science Direct was split into three parts:

- *(SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (optimisation OR optimization OR optimize OR allocate OR allocation)*,
- *(SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (scheduling OR schedule OR "business process" OR reliability OR dependability)*,

- *(SOA OR ESB OR "Service oriented architecture" OR "Enterprise Service Bus") AND (reliable OR dependable OR simulation OR simulate OR genetic).*

We have conducted a search across databases using the following attributes: Title, Keywords, and Abstract. We filtered the results to publications after 2006. In the Springer database, we have chosen the *Computer Science* discipline and the *Information Systems and Applications* sub-discipline. In IEEE Xplore option *Command Search* was selected to use the presented search clause.

The search we have conducted on each database gave the following results:

- ACM – 627 papers,
- IEEE Xplore – 5 334 papers,
- Science Direct – 1 308 papers,
- Springer – 4 491 papers.

### C. PAPERS SELECTION

The paper selection process started with all 11 760 publications. During the first stage of paper selection, titles were read and improper papers were removed from further analysis. If the title did not provide enough information to decide about excluding or including the article (first stage) then the abstract was read. During the second stage, inclusion and exclusion criteria were refined to be precise enough to decide. The publication was being passed into the third stage in case of any doubt in meeting inclusion or exclusion criteria. During the third stage, papers were analyzed more deeply to decide whether to qualify the paper for review. At that stage, 443 articles were analyzed. The analysis consisted of reading the introduction and further chapters, if necessary. At that stage, 123 articles were qualified for the literature review.

The largest number of articles qualified for the review was found in the IEEE Xplore (99) and Springer (10) databases. Table 1 shows the number of articles found in selected databases at the subsequent stages of their qualification for the survey.

**TABLE 1.** The number of papers found in chosen databases.

| Database | 1st stage | 2nd stage | 3rd stage |
|---|---|---|---|
| ACM | 627 | 45 | 7 |
| IEEE Xplore | 5334 | 307 | 99 |
| Science Direct | 1308 | 46 | 7 |
| Springer | 4491 | 45 | 10 |
| **Total** | **11760** | **443** | **123** |

Because of the low amount of articles found in 2017-2019 (5 in 2017, 1 in 2018, and 1 in 2019), we have also applied snowballing technique. In this part of the search process, we have searched through references of previously found articles published in 2020. We have analyzed references to publications of 2017-2019. Out of 390 references, we selected 39 articles for the second stage, and lastly, we have included 5 additional articles in our review. Thus we have found another 2 papers published in 2017, 2 papers published in 2018, and 1 paper published in 2019.

Eventually, 128 articles were qualified for the genuine literature review. Fig. 1 shows the number of articles published in subsequent years covered by the literature review.

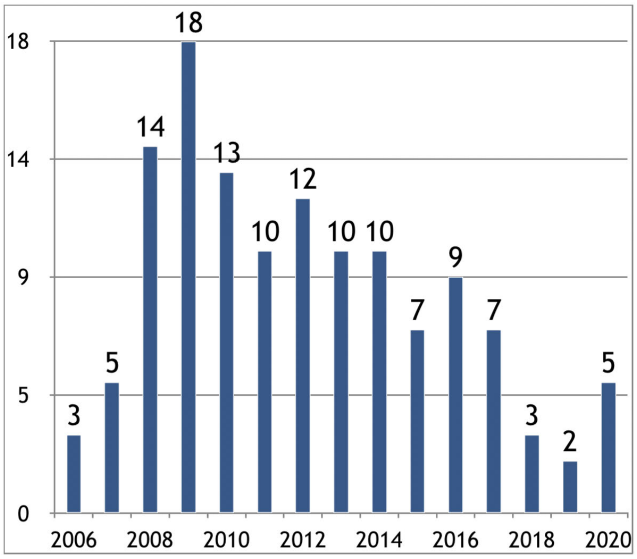


**FIGURE 1.** The number of papers published in years covered by the literature review.

As we can see, the subject is still relevant, but the peak of interest was from 2008 to 2014 years. However, in 2020 we may observe a new increase in the number of papers. The growth of microservices approach popularity is the main cause of that effect. Some articles mention microservices directly, while others point out the rapid increase in the number of services.

### D. QUALITY ASSESSMENT

All works presented within the analysis have been checked by the corresponding author, the senior researcher. The search clause, as well as inclusion and exclusion, criteria have been refined several times during discussions among authors. The search clause during verification has been enriched substantially. Such a modest amount of articles in recent years, especially in 2018 and 2019 has raised our concern. We should bear in mind that optimizing business processes in service systems takes place in the context of classic SOA or microservices. As a result, we have done an additional search for publications in the Science Direct database within the Computer Science category with two sets of keywords, i.e.: *SOA Service Oriented Architecture* and *Microservices*. Fig. 2 shows results for years from 2006 to 2020.

The popularity of classic SOA architecture has been decreasing since 2015, measured with the number of papers within years. However, in recent years we can notice an increase in the number of papers published on microservices.

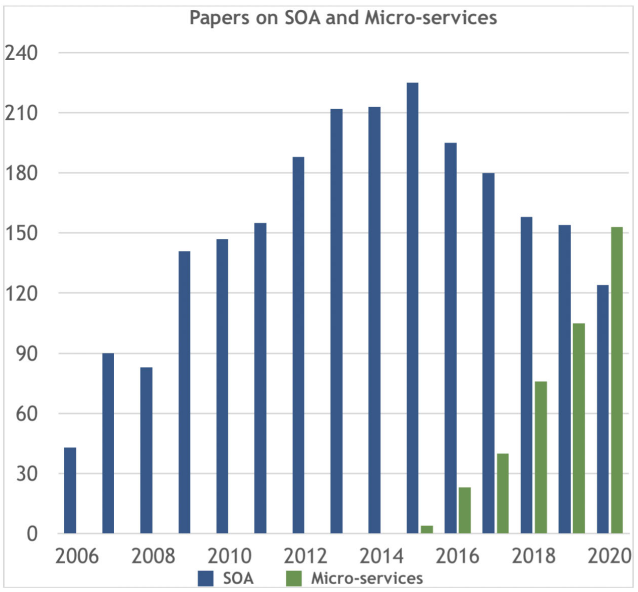


**FIGURE 2.** **The number of papers in the area of service architecture.**

So, the effect we have observed is in line with the trend of changes in the popularity of service architecture types.

### *E. DATA EXTRACTION AND CLASSIFICATION*

All papers at the beginning of the analysis had basic metrics, i.a.: *Authors*, *Title*, *Abstract*, *Year of publication*. We have gained those as a result of search exports from databases.

After selection, we have grouped papers into categories. The main category we have created is *Stage of optimization*. During analysis of the final set of research papers we have found various methods that relate to the following stages of optimization:

- Service Allocation – responsible for selecting components to be deployed on particular servers,
- Service Composition – responsible for selecting component instances that should realize particular steps of business processes,
- Service Scheduling – responsible for determining the order of service tasks execution in a component queue.

The second category we have identified is *Degree of randomness*. According to that category, optimization methods that we have found can be classified into two groups:

- Heuristic – when a heuristic optimization method was proposed in the article,
- Deterministic - when method presented in the article allowed to find an optimal solution.

The next category we have identified is *Degree of reliability*. According to that category, optimization methods that we have found can be classified into two groups:

- Reliability – which means that the method presented in an article considers reliability directly,
- QoS – when the method does not consider reliability directly but allows its user to define reliability within the definition of Quality of Service (QoS).

The last category we have identified is *Time of optimization*. According to that category, optimization methods that we have found can be classified into two groups:

- Online – when a method aims to optimize during run-time and allows to change system behavior every time a new business process is started,
- Offline – when a method is run offline before results are implemented into the production environment.

## III. OPTIMIZATION STAGES

The main aspect considered in the paper is the optimization of the business process implementation in the service architecture. Hence, we focused on discussing the results of the review for individual optimization stages of business process execution.

In order to present those optimization stages, we introduce the required notions:

- A *component* is software that is capable of service execution. A component can provide one or several services. An example of a component could be a Docker container that has a Java program with Representational State Transfer (REST) services implemented.
- *Component deployment* is the process of installing a component on a virtual or physical server. After component deployment, its service is available through WWW.
- *Service* is a provided functionality that can be accessed remotely, usually through WWW. Services are usually implemented using Simple Object Access Protocol (SOAP) or REST. Service is also called web service.
- An *abstract service* is a web service definition that can be implemented by multiple components.
- A *service provider* is a deployed component capable of executing service requests.
- A *service consumer* is a component that uses web service realized by other components to fulfill its purpose.
- A *service request* is an invocation to execute a service made by the service consumer to the service provider.
- A *service instance* is the realization of a single service. Each service can have multiple instances in parallel.
- A *complex service* is a service whose implementation is consisted of multiple other services that should be executed in a specified order.
- A *business process* is an ordered set of tasks aimed at obtaining business value. Usually, business processes within SOA are implemented as complex services. Such business process is available through web service.
- A *business process instance* is a single realization of the business process. One business process can have multiple instances. They can be performed simultaneously.
- A *business process invocation* is a request to perform the business process instance.

Resource allocation is the process of choosing which components should run on which servers. The goal of resource allocation is to select resources for component instances and determine the number of component instances needed to meet performance and reliability requirements. Most often, this process is carried out before the implementation of the system. But some methods introduce changes to the available online resources in response to the current system load. The quality of business process implementation depends not only on how many instances of components that perform services will be implemented. An important aspect is also the deployment of components. For example, components that interact frequently may be deployed at the same server to minimize communication delays. The quality and the costs of process implementation are influenced by the number of resources allocated to individual components. Finally, it is important to determine how many instances of a component will be deployed in parallel on different servers.

Fig. 3 shows UML Deployment diagram with an example of allocating components to servers, where component *Component 2* was deployed on both servers. The component deployed on the server is the service provider. Thus, one provider for *Service 2* (Fig. 3) is *Component 2* deployed on *Server 1* and the other one is *Component 2* deployed on *Server 2*.

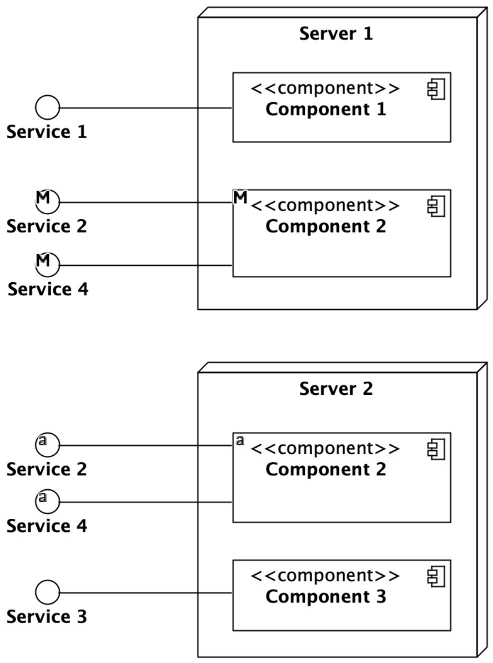


**FIGURE 3.** UML deployment diagram with an example of allocation of components to servers.

At the Service Composition stage, the best instances of services are selected for the implementation of business process steps. The stage occurs when all system components are already deployed, active and information about them is included in the service registry. Business processes are defined along with a set of services that can support them. In the case of many providers which can realize the same service, the selection of provider to run should be made. Each of the services can be performed by one or many providers, as shown in Fig. 2. Each of these providers is characterized by different characteristics, e.g.: execution time, the variance of completion time, probability of successful completion of the task, or accuracy of calculations. In addition, for the same provider, the quality of the different services provided may also be different. It is therefore important choosing the right providers to run particular steps to obtain the best possible result from the entire process point of view. A typical method of composing services requires the input of quality parameters (QoS) of services from different providers. It also requires the definition of business processes as a sequence of services to be invoked. Two more elements should be specified. These are optimization criteria and constraints that are assigned to specific processes (e.g. the process must be performed successfully 99.9% in the shortest possible time). Then, different algorithms are used to find the best combinations of service providers. The most commonly used for this purpose is the genetic algorithm.

Fig. 4 shows an example of a business process whose steps are supported by services implemented by different providers. Thus, the process *Step 2* (Fig. 4) may be performed by two providers.

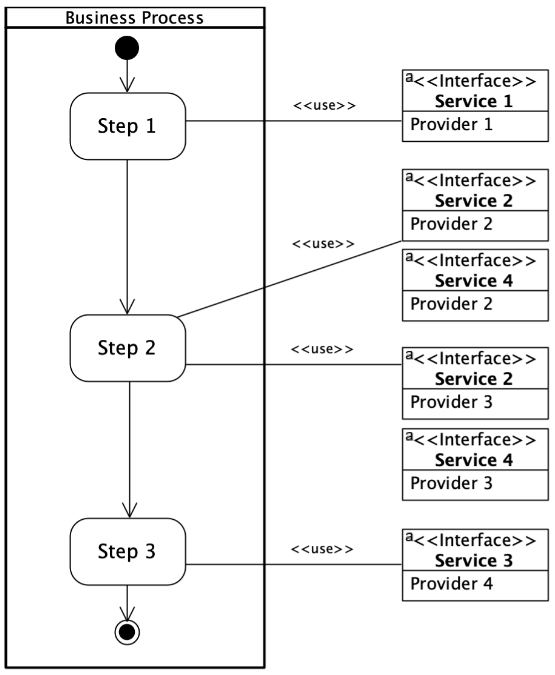


**FIGURE 4.** UML activity diagram with business process steps supported by services realized by various providers.

Service Scheduling takes place during run-time. It is useful when there are many instances of business processes. Steps in such processes invoke services. Services are realized by providers that use components. Requests for service realization are stored in the component's queue. Service Scheduling is determining the order of service request execution in the component queue. Various scheduling policies can be used at queuing component. In practice, the default order is *first-come, first-served* (FCFS). Other scheduling policies may improve the efficiency of a business process. The articles presented in the analysis show that changing the order in which requests are carried out can have a significant impact on improving the efficiency of business processes. For example, increasing the priority of steps on the critical path can reduce the average time to complete the process.

All three types of algorithms apply both in classic SOA and microservices. However, they are implemented in different technologies. In classic SOA at the service composition stage, ESB takes responsibility for dispatching service invocations across components. As far as the service scheduling is concerned, service buses often have their queues and can manage them freely. Resource allocation is usually managed in the deployment phase of the system implementation project. In a microservices architecture, the responsibilities of ESB are split into multiple technologies. Processes can be implemented in components as choreography or orchestrated by dedicated components, e.g.: Activiti, Camunda, or jBPM. Valderas *et al.* [6] show an example of service composition in a microservices architecture. They propose to add a dedicated component called *Global Composition Manager* that executes service composition strategy. They propose to use Eureka as a service registry and RabbitMQ as a message broker. However, it is possible to use alternative microservices technology such as Apache Kafka for the same effect. Both technologies allow managing service queues and change orders. So the same *Global Composition Manager* can execute service scheduling algorithms. Methods to solve the Resource Allocation problem are the same for classic SOA and microservices. However, usually microservices have more components with less functionality that require fewer resources.

## IV. GENETIC ALGORITHMS

As we show later, the most proliferated in the context of business process execution optimization are genetic algorithms. A genetic algorithm is a kind of metaheuristic algorithm inspired by the biological evolution process. A genetic algorithm is a representation of the Darwinian theory of survival of the fittest and was proposed by Holland [7]. The primary elements of genetic algorithm are chromosome representation, fitness function, and biologically-inspired operators.

Table 2 presents the operators used in genetic algorithms grouped in four categories [8].

The standard genetic algorithm consists of the following steps: generation of initial population (many chromosomes are created), execution of fitness function, selection, crossover, and mutation.

The encoding scheme determines how the given information has to be encoded in a particular bit format. Selection determines whether the particular chromosome will participate in the reproduction process or not. Crossover operators are used for generating offspring by combining the genetic information of parents. A mutation is an operator that maintains the genetic diversity between consecutive populations.

The procedure of a genetic algorithm is as follows. A population $C$ of $n$ chromosomes is initialized randomly.

**TABLE 2.** Operators used in genetic algorithms.

| Operator category | Operators |
|---|---|
| Encoding | Binary, Octal, Hexadecimal, Value, Permutation, Tree |
| Selection | Roulette wheel, Rank, Tournament, Boltzmann, Stochastic universal sampling |
| Crossover | Single point, K-Point, Uniform, Partially mapped, Order, Shuffle, Cycle, Reduced Surrogate, Precedence preserving |
| Mutation | Displacement, Inversion, Scramble, Reversing, Bit flipping |

The fitness of each chromosome in $C$ is computed. Two chromosomes $c_1 \in C$ and $c_2 \in C$ are selected from population $C$ according to the fitness value. The crossover operator with crossover probability $P_c$ is applied on $c_1$ and $c_2$ to produce offspring $c_1^o$. Next, a mutation operator is applied on produced offspring $c_1^o$ with mutation probability $P_m$ to generate $c_1'$. The new offspring $c_1'$ is placed in the new population $C'$. The selection, crossover, and mutation operations are repeated on the current population $C$ until the new population $C'$ is complete.

A genetic algorithm is widely used in methods at the Service Composition stage to optimize the selection of service instances. Within those methods, a chromosome (also called genotype) is a selected instance of service for each generic service in the business process. A typical service composition method that uses a genetic algorithm assigns QoS, reliability, or cost parameters to service providers for each service. The most common service parameters are average service execution time, the probability that execution time will exceed a given threshold, the probability of failure during execution (not giving response), and execution cost.

Each business process is modeled, usually as a graph. To use a genetic algorithm, a chromosome is proposed which shows the allocation of steps of business processes to service providers. Each of those genetic algorithm steps is implemented differently in various articles.

## V. RESULTS

We have grouped the results of the literature review into three subsections in the order of optimization stages, i.e.: Resource Allocation, Service Composition, and Service Scheduling.

Table 3 presents the number of articles describing optimization methods broken down into identified optimization stages.

**TABLE 3.** The number of papers describing methods in the optimization stages.

| Optimization stage | Number of papers |
|---|---|
| Service Allocation | 7 |
| Service Composition | 119 |
| Service Scheduling | 3 |

The vast majority of articles are devoted to methods in the Service Composition optimization stage. Fig. 5 shows the percentage of the number of articles related to each optimization stage in the total number of papers.

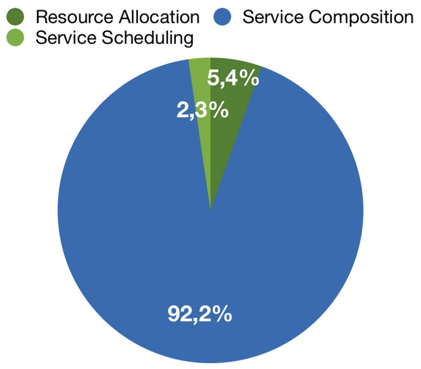


**FIGURE 5. The percentage of papers devoted to particular stages of optimization.**

## A. RESOURCE ALLOCATION METHODS

The first area of optimization of business processes is concentrated around Resource Allocation. When services and processes are set but not yet deployed service allocation needs to be done, which is defining how many instances of each component to deploy and how large resources (e.g. CPU, RAM) need to be assigned. Every concurrent instance of the component increases reliability while adding resources increases performance. On the other hand, it increases costs. The goal is to achieve the best performance while satisfying QoS requirements for the best price or with limited resources. Such a method is presented by Zhang *et al.* in [9] where capacity planning for each service is done for optimal cost while satisfying Service Level Agreement (SLA) requirements. They formalize resource allocation as a capacity planning optimization problem and show the mathematical model and algorithm to solve it. Xie *et al.* in paper [10] focus on the availability of resources of SOA components that are used within the business process. They present Workflow Specification Module that analyzes processes and sets availability requirements for each step on three layers, i.e.: network, application, and database. They also propose Weak-Point Analysis Module and maps requirements with the costs of the increasing availability of each layer. Finally, they propose an algorithm to calculate the optimal allocation of money to elements on every layer for each service. Mennes *et al.* in [11] have used the genetic algorithm in the service allocation problem. They have adapted standard implementation of a genetic algorithm but with the usage of Biased-Random-key as chromosome which consists of three parts, i.e.: order of the applications, number of duplicates of applications, and allocation of duplicates to physical machines. Resource allocation can be also optimized online. Especially if component containers are virtualized and share the same physical resources. Method for such optimization is described by Schmid in [12] where service and workflow managers were proposed. Each component that is responsible for business process orchestration gets a workflow manager. Each instance of the component has a service manager. Workflow managers split the SLA of the business process into many local SLA's for each service included in the process. Service managers monitor the fulfillment of local SLA requirements. If local SLA is violated, then the service manager is responsible to optimize service performance (e.g.: resizing, reallocation of resources, migration of virtual machines). Another optimization method during run-time is presented by Almeida *et al.* in [13] with two parts of optimization. The first part is focused on short-term optimization. With limited resources, service instance workload is predicted and then the allocation is made based on predicted workload. Prediction is based on the pay and penalties mechanism. Each QoS validation gets a penalty while each realization within QoS restriction gives pay. The authors present a mathematical model based on this mechanism and the Fixed Point of Iteration algorithm, which is to optimize global revenue. Such short-term optimization should be executed every 10–30 minutes. The second part is long-term optimization, which aims is to set an optimal amount of physical resources, however, it is not elaborated in the article. Abdullah *et al.* in paper [14] also present a method to optimize the allocation of resources but only for containers running on the same machine.

Both areas of Resource Allocation and Service Scheduling are the subject of the solution proposed by Huang *et al.* in [15]. Resource allocation starts with generating Service Relationship Graph based on business processes. Then it applies a minimum *K-Cut* algorithm based on a greedy strategy to find the lowest amount of servers needed to satisfy performance and reliability requirements. In the service scheduling area, the bottom amount rank algorithm is proposed. It calculates the weight of every step for every business process. Weight is higher for service invocations that have more steps after it. So the longer the business process and earlier step, the more weight it is going to have. Then two rules are applied. First requests with higher ranks will be processed first if they are within the same process. Second requests with lower ranks will be executed if they are from different processes.

## B. SERVICE COMPOSITION METHODS

The most popular in optimization of business processes in services architecture is Service Composition that takes into account QoS parameters. The typical Service Composition method requires at start information about service instances such as response time, and QoS (e.g. measured as the chance of not responding correctly in required time). In a standard Service Composition method, every business process or complex service (both concepts are used, but they mean the same thing in Service Composition) comprises multiple services that have to be realized in a specified order. Each service is realized by a component, which is technologically independent software. There have to be many components or many deployments of the same component able to realize the same service for Service Composition to make sense. The purpose of Service Composition is to select components to realize services in such a way that business process will work optimally, e.g. will be realized fastest at a cost or will be realized cheapest while not exceeding a specified time. Service Composition methods can be divided into two groups in the category *Time of optimization*. The first group encompasses *Online* methods that are used during run-time. Those methods can give different results for every business process instance. That means that every instance of the business process may invoke different components to realize the same services. They react to changing parameters of the systems, e.g.: number of service instances, the performance of services, the density of business processes invocations. The second group consists of *Offline* methods that are intended to be used by developers before deployment. Those methods expect programmers to choose instances from a set of results (e.g. Pareto optimal results). Fig. 6 shows the most used algorithms at the Service Composition stage.

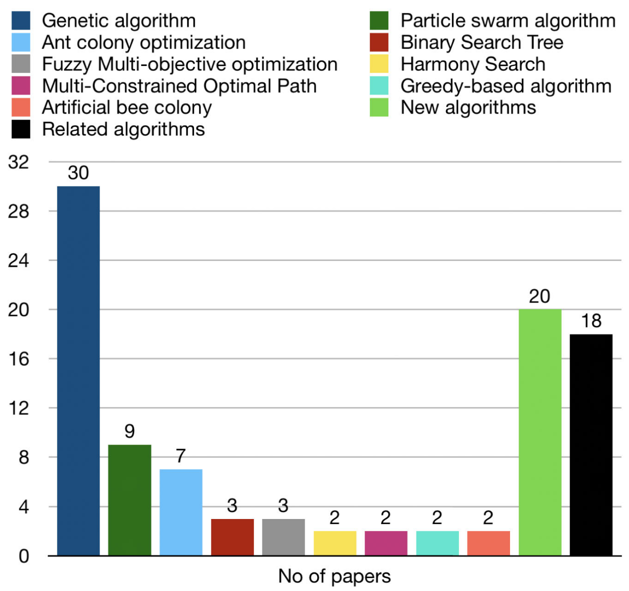


**FIGURE 6.** **Number of papers using selected algorithms at the service composition stage.**

Because most of the papers are devoted to the use of genetic algorithms, we discussed these in the first subsection. Next, we have presented papers that apply other known algorithms. As we can see in Fig. 6 there is a significant number of new, authorial approaches to Service Composition (20). We have introduced them in the consecutive subsection. There is also a large number of papers that are indirectly related to Service Composition (18), e.g.: concentrate on architectural aspects of services. Those papers we have described in the last subsection.

### 1) GENETIC ALGORITHMS

Most Service Composition methods use a genetic algorithm to optimize the selection of service instances. We have found the following papers that consider using an evolutionary algorithm to optimize service composition: [16]– [42], [44]– [47]. Some of them implement the previously presented standard scenario with little modifications, while others introduce some greater extensions. So further, we present methods that differ from that standard scenario. Method presented by Ludwig [20] proposes unique selection step (clonal selection). Ebrahim [21] enrich standard genetic algorithms with provider limits and they minimize costs. Seghir and Khababa [134] also show a method with a unique selection step of a genetic algorithm. This time authors want to increase the diversity of results. To achieve this, they improve selection probability to the next generation of those results that have more differences from the others. Simultaneously, they include the standard fitness function to determine the survival chance of genotype. They also split optimization into two steps where in the first step they use a genetic algorithm, and in the second fruit fly algorithm. Another example of evolution optimization is proposed by Wang *et al.* in [132]. They propose unique genotype solution notation based on trees. da Silva *et al.* [25] use directed acyclic graphs as genotypes to increase computational efficiency. Papers [26], [33], and [34] present methods which results are Pareto-optimal for developer to select. Radwan *et al.* in [28] presented a method that requires a listing of all risks and their impact and then they are optimized using the standard method. In [29] Klein *et al.* interpret QoS as time constraints and their method considers only time on network delay. Zhou *et al.* [30] show that genetic algorithm is most efficient in Service Composition problem and they extend algorithm with differential evolution. Their algorithm enriches the standard approach with functional correctness check which is input and output comparison between generic service and service instance. Klein *et al.* [35] present a standard Service Composition method for B2B purposes which is optimized to handle a greater number of process instances and higher reliability requirements. Both papers [36] and [37] show a method that is optimized to respond to changes of instances (new instances can show up while old ones can be stopped). Li and Li [39] use Nash Equilibrium theory and sets it as objective (Nash Equilibrium between service providers and consumers) which is achieved by standard means (Genetic algorithm). In paper [40] two dimensions are optimized: quantity (such as reaction times) and security (e.g. level of data protection). Two algorithms are proposed to optimize those dimensions: genetic and global optimization algorithms with local selection. Zhao *et al.* [41] show Service Composition that uses Chaos Genetic Algorithm where chaos element is the migration of population so local extremes will be easier to exit and convergence will be achieved faster. Method in paper [42] extends service composition problem with backup scenarios. Backup scenarios are cases in business processes where a service instance fails to achieve its purpose, and some steps have to be performed to undo changes already made. A genetic algorithm is used, which presents Pareto-optimal solutions to a developer who has to pick one before implementation.

### 2) KNOWN ALGORITHMS

Many methods fit in the scheme presented before but use other widely known algorithms instead of genetic ones. A Particle swarm algorithm is also used to solve the Service Composition problem. It can be found that Zhong *et al.* in [53] merge some service providers into coalitions and then compare such coalitions and different combinations of them. Particle swarm algorithm is also used by Wang *et al.* in [54] where service search and practical implementation propositions are also presented. The same algorithm is also proposed in papers [55]– [60], [130].

Table 4 presents other known algorithms, used at the Service Composition stage, that have been found in the papers included in the review.

An example of another type of algorithm is presented, by Qiqing *et al.*, in [48] where a directed graph is used to represent a business process. Each service has parameters such as price, execution time, reliability. They use an Ant colony optimization algorithm to find the optimal solution. Other methods that uses same ant colony optimization algorithm are presented in: [49]– [51], [125], [129]. MinMax Ant colony algorithm combined with culture algorithm is used by Liu *et al.* in [52]. Another method that is based on nature observations is proposed by Gavvala *et al.* in [135]. This time two algorithms are used in sequence. First for wide exploration based on eagle search. Then whale optimization algorithm is used to find the optimal solution in a reduced area. Furthermore, as far as greedy-based algorithms are concerned, Parejo *et al.* [81] show a combination of Greedy randomized adaptive procedure with Path re-linking. Furthermore, Wu *et al.* [82] have also presented a greedy-based algorithm but that time combined with prediction methods. Fuzzy logic for Service Composition has been also presented by Bakhshi *et al.* in [86]. However, the authors do not specify the algorithm they use. Instead, they show how to model that problem.

**TABLE 4. Known algorithms used at the service composition stage.**

| Name of algorithm | Reference |
|---|---|
| Particle swarm algorithm | Zhong et al. [53], Wang et al. [54], Yu et al. [55], Li et al. [56], Liao et al. [57], Xia and Li [58], Li and Li [59], Fan et al. [60], Zhao et al. [130] |
| Ant colony optimization | Qiqing et al. [48], Chen et al. [49], Zhang et al. [50], Wang et al. [51], Liu et al. [52], Alayed et al. [125], Allali et al. [129] |
| Binary Search Tree | Oh et al. [73], Chan et al. [74], Li et al. [75] |
| Fuzzy Multi-objective optimization | Tan et al. [85], Palii and Novogrudska [126], Viriyasitavat and Bi [131] |
| Harmony Search | Mohammed et al. [61], Esfahani et al. [62] |
| Multi-Constrained Optimal Path | Huang et al. [79], Luo et al. [80] |
| Greedy-based algorithm | Parejo et al. [81], Wu et al. [82] |
| Artificial bee colony | Liu et al. [63], Dahan et al. [133] |
| Backward Search | Mohr et al. [64] |
| Differential evolution | Pop et al. [65] |
| Fruit fly optimization | Zhang et al. [66] |
| Bat algorithm | Hashmi et al. [67] |
| Group leader algorithm | Xiang et al. [68] |
| Learning Depth First Search | Nam et al. [69] |
| Negative selection | Zhao et al. [70] |
| Markov Decision Process | Bannazadeh et al. [71] |
| Blackboard system approach | ul Haq et al. [72] |
| Learning Automation | Zhang et al. [76] |
| Bayes theorem | Chen et al. [77] |
| Artificial neural network | Zhang et al. [78] |
| Jensen's inequality | Menascé et al. [83], Menascé et al. [84] |
| Cuckoo search algorithm | Ghobaei-Arani et al. [136] |

### 3) NEW, ORIGINAL ALGORITHMS

There are also approaches where the classical service composition problem is solved with new original heuristic algorithms. Such methods can be found in the following papers [87]–[94]. Another algorithm based on Markov Chains is presented by Hwang *et al.* in paper [95], which minimizes the probability of failure. The approach to maximize reliability is presented by Xi *et al.* [96] where a simple deterministic algorithm is proposed. Guidara *et al.* [97] group up service providers that have similar QoS attributes, and then they reject those that do not meet QoS requirements. Among those that were not rejected, they pick Pareto optimal solutions with their algorithm. Algorithms based on the grouping phase are presented by Deng and Xing [98] and Xia *et al.* [99]. Another two papers, written by Liu *et al.* [100] and by Li *et al.* [101], present two original heuristic algorithms, the first paper for local, and the latter for global optimization. Schuller *et al.* [102] present a method for complex workflows (compositions) where they aggregate QoS values to simplify the problem and then they formulate a linear optimization problem without proposition of the best algorithm to find a solution. Moreover, Živkovic *et al.* [103] present an approach with a simple algorithm where each validation of QoS is a cost, and every time QoS is met some earns are assigned. The method optimizes service selection by choosing services with the highest revenue. In the paper, [104] Wang *et al.* propose a method, where service composition is optimized during operation (online). Based on pre-defined QoS requirements for composite service and current server load. However, it does not show an algorithm for selecting services. The paper only presents how to parse requirements and split them between service components. Ahmed and Srivastava [105] show an algorithm to optimize service composition based on the physical model of friction. Whereas, Li *et al.* [106] present their method that was split into two phases. First is the local optimization algorithm that for each service type predicts the performance of component instance and then selects optimal component. The second phase is based on the first one and its goal is to compose processes by selecting instances. The major part of this method is the prediction of QoS parameters for each service. Finally, Jiang and Bai [107] describe another heuristic method. The method groups service instances on the basis of their historical QoS correlations and then selects the most trustworthy group.

#### 4) RELATED ALGORITHMS

We have also found papers that relate to Service Composition but do not focus directly on optimizing the efficiency of algorithms. Those papers expand the view on the problem. As an example of such an algorithm, we can give that shown by Lee *et al.* [108] where the focus is on constructing alternative solutions in case of the main solution does not work. Depending on the business process attributes (like criticality) it proposes automatic mechanisms to handle situations of main service failure (e.g.: retry sending the request, ignore current step). Wu *et al.* [43] present a method that differs from those above in a way that extends the spectrum of acceptable solutions. In other articles, the business process was set, and it was unchangeable. In this method, the authors show that different generic services can be used to achieve business process goals. Some services can be aggregated or decomposed. For example, a complex service that comprises three steps: checking weather, translates it into the language of the user, and send it via SMS gateway can be realized by three service instances that perfectly match a sequence of steps. But it can be also realized by two service instances: one that gives weather in the selected language and the second one that sends SMS. The presented method analyzes inputs and outputs of service instances and extends accepted solutions which are then used in the genetic algorithm to select the near-optimal one. A similar approach is presented by Kalasapur *et al.* in [109] where authors propose to define business process with basic services (building blocks) and then algorithm searches thru service instances that realize one or several building blocks.

Another way to expand the spectrum of solutions is presented by Ukor and Carpenter in [110] where the extra dimension is added. For every abstract service, it searches through, many finer ones and checks if composite service can be constructed to realize abstract service in the process. Tao *et al.* [111] present a method that is specialized to optimize the service composition of services provided by hardware. They invent an algorithm that is based on two steps: search and decision making. To achieve both, there are several complex algorithms presented. The architecture proposal of a framework that is capable to implement service composition was described by Oppong and Khaddaj in [112]. From the architecture point of view, they showed how to organize elements such as QoS Processor, Service Composition, Resource and Service lit, and Service discovery. The architecture was also a major subject in [113] where Fdihla *et al.* presented a decentralized service composition approach. Usually, service composition is realized by Enterprise Service Bus or BPM Engine (e.g. Activiti) which are both centralized approaches. The way to decentralize service composition is presented by Peixoto *et al.* in [114] where a hierarchical model is proposed. Services are grouped and schedulers are used to organize service composition within the group. The method allows every scheduler to use a different Service Composition optimization algorithm. Another paper with an architecture proposal was published by Bian *et al.* [115]. It presents roles that components should execute within Workflow-SOA architecture and the algorithm for the execution of workflows within those components. Moreover, optimization of Service Composition but on the logical organization of code is presented by Kathiravelu *et al.* in [116]. In that paper, the authors proposed a manager of operations like Find, Invoke, and Return.

In [117] Andersson *et al.* present a framework for run-time service composition. In this framework, the original Service Composition optimization algorithm is presented together with a proposition on how to monitor service providers' performance during and selection rules during the system's life cycle to update selected service instances. The solution to a similar problem of monitoring and reacting to changes is presented by Guidara *et al.* in [118]. Ukor and Carpenter in [119] state that business processes can have many execution paths, e.g.: every XOR gateway splits the process into multiple such paths. Every execution path can have a distinct set of optimal service instances. Authors propose to perform service composition for each execution path and during the execution of the business process pick one path at the beginning based on some initial data. A similar approach of splitting the business process into execution paths is presented by Ramacher and Monch in [120] except that in this one reliability on QoS is optimized and changed every time information about service instance changes. In [121] Dong and Jiao present methods on how to achieve SLA level during Service Composition optimization. On the other hand, Wang *et al.* in [128] show a method how to plan reactions to failures during execution.

Another service composition method is presented by El Haddad *et al.* in [122], where the transactional aspect is most important. It analyzes transactional capabilities of services, like the ability, to undo or to forward recovery, and based on those capabilities it proposes the best solution that is transactional and optimal in the QoS sense. In the paper, [127] Takahashi *et al.* describe a method for the selection of service composition algorithm. It proposes the best algorithm based on user preferences. The preferences have been modeled as QoS requirements that fall into two groups: Essential QoS (must consider), Priority QoS (should consider). Then it compares the algorithm's characteristics with QoS requirements and proposes the best service composition algorithm.

### C. SERVICE SCHEDULING METHODS

Service Scheduling algorithms are used when there are many service requests waiting for realization in a queue of the service instance. Especially, when those requests are parts of many business processes then Service Scheduling can take place. It sets realization order in queues of service instances so overall business process performance is optimized. It may be a real advantage if business processes are of a different value for the company. Normally, without the Service Scheduling method service requests are realized in FCFS order. In some cases, a time-sharing strategy is used or a combination of FCFS and time-sharing.

A simple service scheduling method is presented by Dyachuk and Deters in [123]. The authors propose to select steps that are on a critical path, which has an effect on the overall process. Then give those selected requests higher priority than others and realize in the queue first.

A more complex approach is proposed by Dyachuk and Deters in [124] with two layers of scheduling: local for managing priority in queue and global for setting rules for local schedulers. The global scheduling step sets QoS requirements for each step of the workflow by splitting global QoS into every step. It is done with the Proportional Fraction policy, which is based on historical data about the average response time of each step. Then local scheduling takes place according to one of four presented policies: Highest Value First, Earliest Deadline First, Highest Ratio First, Lawler's Scheduling Method. Highest Ratio First is a combination of two first policies. Authors assume that components have a queue of requests while realizing several in parallel.

## VI. GENETIC ALGORITHMS ANALYSIS

In the optimization of business process execution, algorithms from three groups are used: heuristic, graph, mathematical/numerical. Heuristic algorithms are the most proliferated in that area of optimization. Fig. 7 shows a taxonomy that we have proposed for heuristic algorithms in the review.

In the review, as many as 32 articles use the genetic algorithm to optimize the business process execution. Therefore, in the remainder of this section, we compared the properties of the genetic algorithms proposed in those selected articles. We looked through those papers and extracted the distinguishing features of proposed genetic algorithms. Moreover, we have enclosed advantages of each approach. It is extremely hard to show disadvantages because the authors have not mentioned them in their papers. Besides, such a detailed analysis of each case is beyond the scope of that work. As a result, we presented the contribution and benefits of each work in the following figures. Fig. 8 shows the contribution and benefits of the first eight papers.

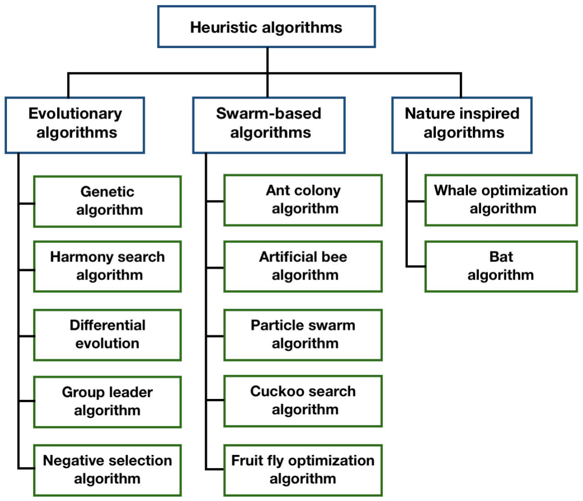
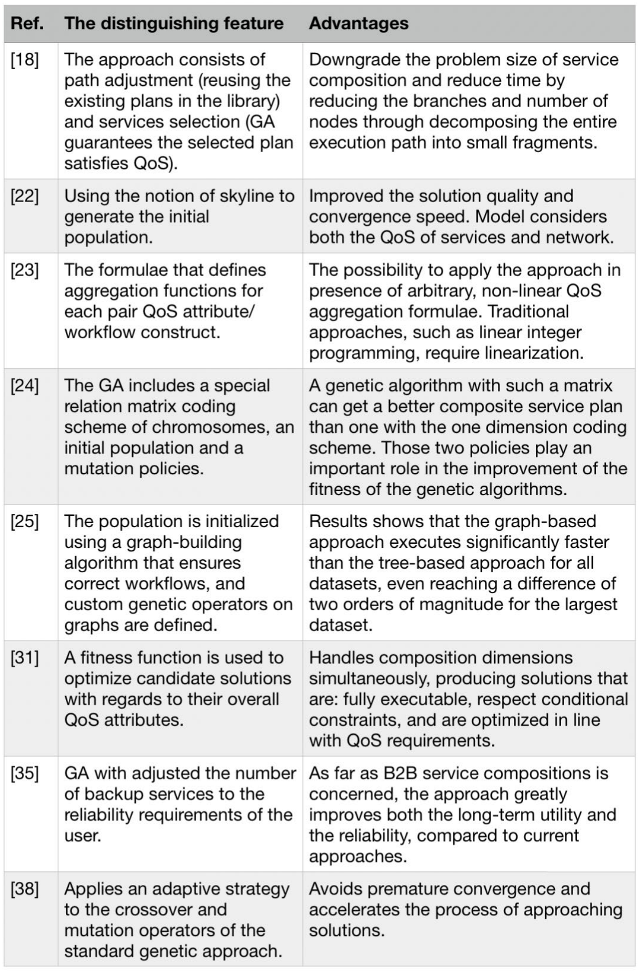


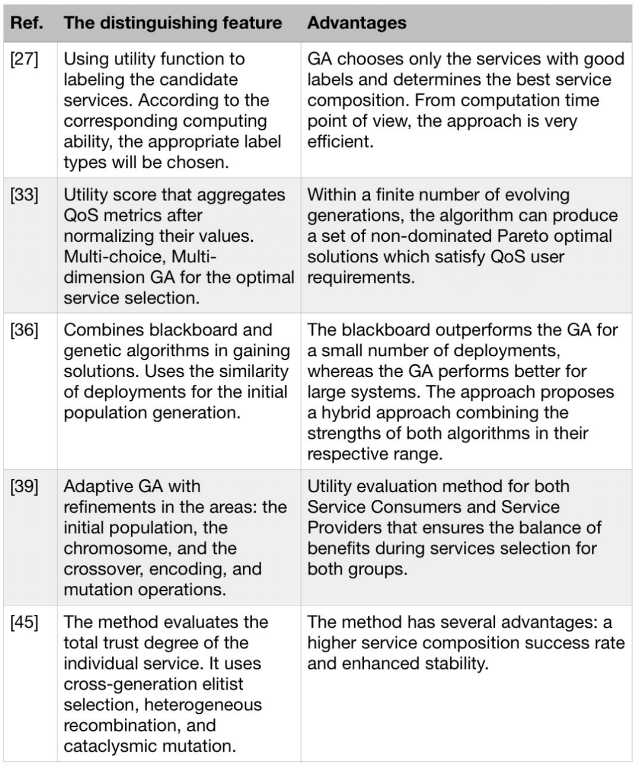
FIGURE 7. Taxonomy for heuristic algorithms in the review.



| Ref. | The distinguishing feature | Advantages |
|---|---|---|
| [18] | The approach consists of path adjustment (reusing the existing plans in the library) and services selection (GA guarantees the selected plan satisfies QoS). | Downgrade the problem size of service composition and reduce time by reducing the branches and number of nodes through decomposing the entire execution path into small fragments. |
| [22] | Using the notion of skyline to generate the initial population. | Improved the solution quality and convergence speed. Model considers both the QoS of services and network. |
| [23] | The formulae that defines aggregation functions for each pair QoS attribute/ workflow construct. | The possibility to apply the approach in presence of arbitrary, non-linear QoS aggregation formulae. Traditional approaches, such as linear integer programming, require linearization. |
| [24] | The GA includes a special relation matrix coding scheme of chromosomes, an initial population and a mutation policies. | A genetic algorithm with such a matrix can get a better composite service plan than one with the one dimension coding scheme. Those two policies play an important role in the improvement of the fitness of the genetic algorithms. |
| [25] | The population is initialized using a graph-building algorithm that ensures correct workflows, and custom genetic operators on graphs are defined. | Results shows that the graph-based approach executes significantly faster than the tree-based approach for all datasets, even reaching a difference of two orders of magnitude for the largest dataset. |
| [31] | A fitness function is used to optimize candidate solutions with regards to their overall QoS attributes. | Handles composition dimensions simultaneously, producing solutions that are: fully executable, respect conditional constraints, and are optimized in line with QoS requirements. |
| [35] | GA with adjusted the number of backup services to the reliability requirements of the user. | As far as B2B service compositions is concerned, the approach greatly improves both the long-term utility and the reliability, compared to current approaches. |
| [38] | Applies an adaptive strategy to the crossover and mutation operators of the standard genetic approach. | Avoids premature convergence and accelerates the process of approaching solutions. |

FIGURE 8. Contribution and benefits of genetic algorithms (1–8).

The authors of selected papers touch on various elements of genetic algorithms. They deal with initial population, fitness function, and crossover and mutation operators. Authors use graphs to decrease the complexity of service composition by reducing the number of branches and nodes. Results reveal that the graph-based approaches execute significantly faster than the tree-based ones. The next figure (Fig. 9) shows the contribution and benefits of consecutive five papers.



| Ref. | The distinguishing feature | Advantages |
|---|---|---|
| [27] | Using utility function to labeling the candidate services. According to the corresponding computing ability, the appropriate label types will be chosen. | GA chooses only the services with good labels and determines the best service composition. From computation time point of view, the approach is very efficient. |
| [33] | Utility score that aggregates QoS metrics after normalizing their values. Multi-choice, Multi-dimension GA for the optimal service selection. | Within a finite number of evolving generations, the algorithm can produce a set of non-dominated Pareto optimal solutions which satisfy QoS user requirements. |
| [36] | Combines blackboard and genetic algorithms in gaining solutions. Uses the similarity of deployments for the initial population generation. | The blackboard outperforms the GA for a small number of deployments, whereas the GA performs better for large systems. The approach proposes a hybrid approach combining the strengths of both algorithms in their respective range. |
| [39] | Adaptive GA with refinements in the areas: the initial population, the chromosome, and the crossover, encoding, and mutation operations. | Utility evaluation method for both Service Consumers and Service Providers that ensures the balance of benefits during services selection for both groups. |
| [45] | The method evaluates the total trust degree of the individual service. It uses cross-generation elitist selection, heterogeneous recombination, and cataclysmic mutation. | The method has several advantages: a higher service composition success rate and enhanced stability. |

**FIGURE 9. Contribution and benefits of genetic algorithms (9–13).**

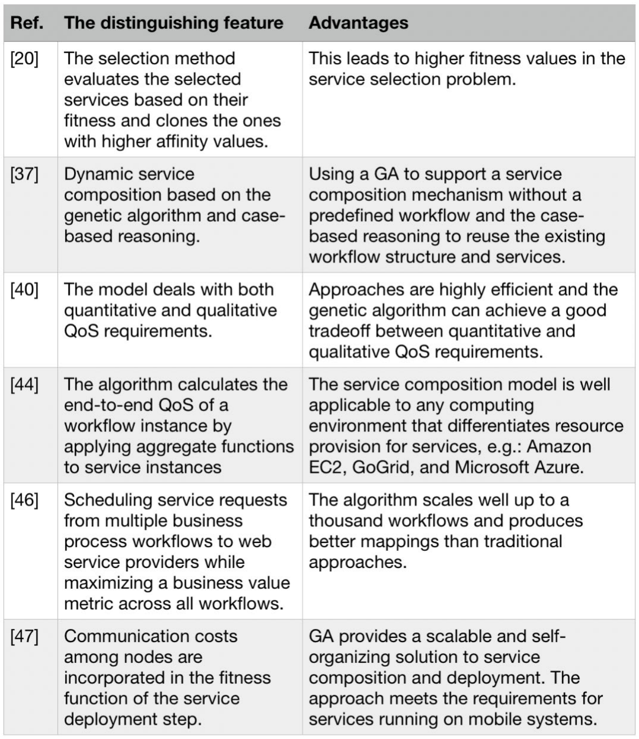
Fig. 10 summarizes the next six papers.

We can see that still, the fitness function is in the area of interest of researchers. They strive for the optimal qualities of individuals. Works offer also refinements in the areas: the initial population, the chromosome, and the crossover, encoding, and mutation operators. Moreover, researchers concentrate on proposing scalable solutions that meet QoS requirements.

Fig. 11 shows the contribution and benefits of the following eight papers.

The authors concentrate on solving service composition optimization while fulfilling QoS requirements, in some cases contradictory. They strive for better adjustment of GA to solving the optimization problem. Some of them identified the preferable parameters for its mutation rate, number of generations, the penalty factor for constraint-missing individuals, and the implementation of the fitness function. It worth emphasizing that some authors developed algorithms that can reach a suboptimal solution within 8% of the time required for getting the optimal solution. It reduces the time needed to obtain an acceptable solution by an order of magnitude. It also shows that it is worth investing effort in the development of this type of algorithms.

Fig. 12 shows the contribution and benefits of the last five papers.



| Ref. | The distinguishing feature | Advantages |
|---|---|---|
| [20] | The selection method evaluates the selected services based on their fitness and clones the ones with higher affinity values. | This leads to higher fitness values in the service selection problem. |
| [37] | Dynamic service composition based on the genetic algorithm and case-based reasoning. | Using a GA to support a service composition mechanism without a predefined workflow and the case-based reasoning to reuse the existing workflow structure and services. |
| [40] | The model deals with both quantitative and qualitative QoS requirements. | Approaches are highly efficient and the genetic algorithm can achieve a good tradeoff between quantitative and qualitative QoS requirements. |
| [44] | The algorithm calculates the end-to-end QoS of a workflow instance by applying aggregate functions to service instances | The service composition model is well applicable to any computing environment that differentiates resource provision for services, e.g.: Amazon EC2, GoGrid, and Microsoft Azure. |
| [46] | Scheduling service requests from multiple business process workflows to web service providers while maximizing a business value metric across all workflows. | The algorithm scales well up to a thousand workflows and produces better mappings than traditional approaches. |
| [47] | Communication costs among nodes are incorporated in the fitness function of the service deployment step. | GA provides a scalable and self-organizing solution to service composition and deployment. The approach meets the requirements for services running on mobile systems. |

**FIGURE 10. Contribution and benefits of genetic algorithms (14–19).**

Authors try to consider various aspects of users' requirements. Hence, the proposal of using different fitness functions to address important requirements types. Moreover, there is one application of a genetic algorithm to resource allocation optimization. As far as execution time is concerned the results are more than promising.

## VII. DISCUSSION AND LIMITATIONS

In the section, we attempt to answer research questions, share our insight, and highlight the limitations of the study. To answer RQ1 and RQ4 research questions, we grouped articles into three categories: Resource Allocation, Service Composition, and Service Scheduling. The Service Composition category is the most elaborated one. There are 119 articles in this area which make up the majority of the articles reviewed. Most of them are very similar in foundations but differ in the algorithms used to search for solutions. We could find there both the reuse of known algorithms and proposals for new ones dedicated to the described problem. There is a large variety of algorithms used. Most popular is a genetic algorithm with 32 methods that use it. However, there were a lot of attempts to use other algorithms that achieved better results under specific conditions. The area of Service Composition is very well elaborated, and it is very hard to find weak spots. The only thing in our opinion is missing is a method that could combine all three areas which we will elaborate on further.

The second area is Service Scheduling in which we only found 3 articles. The most used scheduling strategies in articles about Service Composition are FCFS and time-sharing.

| Ref. | The distinguishing feature | Advantages |
|---|---|---|
| [16] | A combined model with both data distribution for parallel computing by parallel tasks and selection of services that meets QoS constraints. | The solution is able to select services in such a way that the execution time is minimized and the total cost of selected services does not exceed the budget. |
| [17] | Approach for automated service composition that constructs QoS-aware and optimal service composition with users' constraints. | Reduces the composing time and addresses the scalability of service composition. |
| [21] | GA takes into consideration the dynamic cost of the services and the constraints imposed on the service providers. | The algorithm is able to reach a suboptimal solution within 8% of the time required for getting the optimal solution. |
| [26] | GA is employed to find out the optimal combinations of Web APIs in a composite service flow. | Results show that the approach can effectively locate the combination of Web APIs based on the user-preferred QoS indicators while preserving other properties of service. |
| [30] | New difference evolution to improve the quality of individuals. | Compared to E3-MOGA the algorithm has better efficiency in finding all feasible individuals of higher QoS. |
| [32] | Selecting services while considering different QoS characteristics. | Identified the preferable parameters for its mutation rate, number of generations, the penalty factor for constraint-missing individuals, and the implementation of the fitness function. |
| [34] | An optimization framework to address the QoS-aware service composition problem. | Defines a service composition model and provides a multiobjective genetic algorithm E3-MOGA to solve the QoS-aware service composition problem. |
| [41] | Chaos theory is used to generate the initial population of the genetic algorithm. | Results show that the approach for Web service composition considering the constraints between services has better performance. |

**FIGURE 11.** **Contribution and benefits of genetic algorithms (20–27).**

| Ref. | The distinguishing feature | Advantages |
|---|---|---|
| [11] | Distributed GA places service-oriented applications on a hybrid cloud, by defining a representation of an application placement in a biased-random-key chromosome and using a fault-tolerance distributed pool model. | Results show that for high availability requirements, the execution time is up to 1000 times faster with only a very small decline in the performance of the placement ratio. |
| [19] | GA has four fitness functions to score chromosomes from diverse aspects. Each fitness function has a priority order. | GA satisfies the user's functional requirements, QoS criteria, and transactional requirements at the same time. |
| [28] | Adapted the DREAD model for security risk assessment. New categorizations for calculating factors based on a proposed service structure and service attributes. | The tool adds security services to service composition, which reduces the risk severity of the generated composition and enhances its security in terms of confidentiality, integrity, and availability. |
| [29] | A network-aware approach that handles the QoS of services and the QoS of the network independently. | A self-adaptive genetic algorithm that balances the optimization of latency and other QoS as needed and improves the convergence speed. |
| [42] | The multi-objective approach considers the possible repair costs directly in the initial service composition. | A new QoS model that helps to predict the resulting QoS of a workflow by considering service failures during the initial selection phase. |

**FIGURE 12.** **Contribution and benefits of genetic algorithms (28–32).**

Both are used most commonly in practice. It is very hard to implement the Service Scheduling method because it requires monitoring of the status of all processes and component instances and algorithms that can provide results fast enough to give value. However, monitoring techniques are developing rapidly, especially in the microservice implementation of SOA. Therefore, there is great potential for further research. Service composition articles inspire aspects that could be optimized using service scheduling. Among others, we can name aspects: prioritization according to the business value of business process, prioritization according to QoS or reliability requirements. Such optimization could allow achieving the same QoS or business value with fewer resources.

The last of the analyzed stages of business process optimization is Resource Allocation. We have found only 7 articles about this subject. A common feature of those articles was a significant limitation of those methods' application. For example, Zhang *et al.* [9] focused only on minimizing the probability of failure, Xie *et al.* [10] narrowed its optimization to availability, while Abdullah *et al.* [14] focus on allocation only within one machine which runs many containers. This means that these methods can only be used under specific circumstances. We argue that during the deployment phase of the SOA system usually many aspects should be considered simultaneously such as realization time and QoS. Therefore, we see potential in this area for further exploration.

There are much more articles that propose online methods (57) than offline ones (19). But, many can be used in both manners. Fig. 13 shows percentage breakdown of the number of articles published in groups of *Online* and *Offline* methods of *Time of optimization* category.

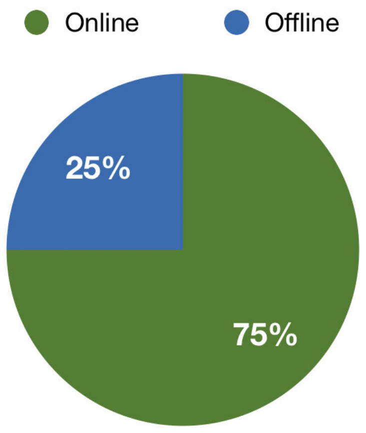


**FIGURE 13.** **Percentage of papers that relate to online and offline methods.**

As far as the RQ2 research question is concerned, there are multiple different answers to the fallibility problem. Most of the articles include fallibility as one of the QoS or SLA aspects. Some consider it separately or even put it as a major topic of the publication. For example, Takahashi *et al.* [127] present a method to find an optimal composition in case of service failure.

Fig. 14 shows percentage breakdown of the number of papers published in groups of *Reliability* (17) and *QoS* (113) of *Degree of reliability* category.

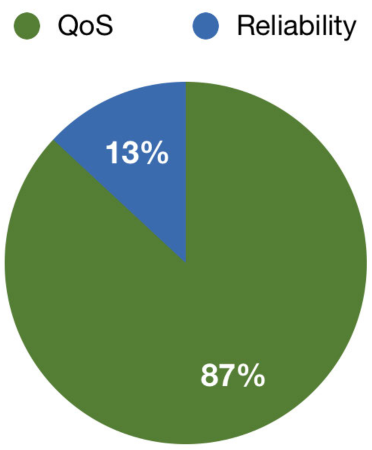


**FIGURE 14. Percentage of papers published in groups of reliability and QoS.**

As an answer to the RQ3 research question, we can conclude, based on the review, that heuristic algorithms are the dominant tool to optimize business processes in SOA. Multiple types of algorithms can be found to solve the problem. Most popular is the genetic algorithm, but there are other commonly used which are: Ant colony, Particle swarm, and Harmony search. Fig. 15 shows percentage breakdown of the number of papers published in groups of *Heuristic* (119 papers) and *Deterministic* (6 papers) methods of *Degree of randomness* category.

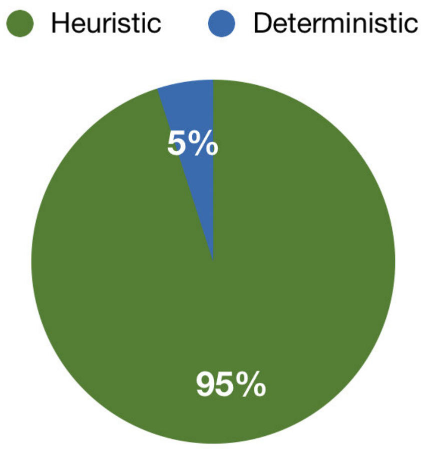


**FIGURE 15. Percentage of papers published in degree of randomness category.**

As far as the RQ4 research question is concerned, the optimal execution of a business process is a very complex issue, and heuristic algorithms yield better results with fewer simplifications. The problem of service composition is NP-hard with $n^m$ possible solutions where $m$ is the number of services in processes and $n$ is the number of service providers [135]. Hence the popularity of heuristic algorithms to solve this problem. Deterministic algorithms are much less popular because they require the introduction of strong assumptions limiting their applicability. For example, Schuller *et al.* [102] propose a method based on linear programming. However, they optimize only those steps that are on the critical path, which reduces the process to the linear one. For tasks that are not on the critical path, they suggest simply choose the cheapest service provider. Another example of the usage of a deterministic algorithm for a limited-service composition problem is shown by Lee *et al.* [108]. In that paper, the authors propose a Service Scheduling method that focuses on finding solutions in case of failure of the service provider during execution.

All those kinds of methods influence each other. Different methods of Service Scheduling and Service Composition may be optimal for different resource allocations. No article was found that proposed to search through available methods and propose optimal algorithms that complement each other at all three stages. Such an approach could widen the area of optimization and give better results than choosing methods at each stage separately.

The problem of service composition is NP-hard with exponential complexity. Therefore, reviewing all possible solutions to select the optimal one is definitely not an efficient way. There are two approaches to deal with the complexity. Firstly, we can simplify the problem as it is done in the deterministic optimization methods. Secondly, we can use heuristic algorithms to solve the problem. In that manner, we reduce the number of examined solutions at the cost of finding a suboptimal one. The latter is the dominant way. Heuristic algorithms usually have polynomial complexity, e.g., genetic algorithm, Ant colony. Moreover, the detailed differences in complexity between the algorithms largely result from their implementation. Two main aspects of the complexity of an algorithm, according to Donald Knuth, are the frequency of executing each statement and memory usage that heavily depends on applied programming statements and data structures. It may have a tremendous impact on the effectiveness of a solution. However, it is hard to compare methods to find those differences. It would require the implementation of those methods and using multiple representative optimization examples to compare. Many of the cited papers showed some performance results but done with a limited number of compared methods. Usually, the conclusion was that the proposed method was more efficient than the others. Implementation and comparison of the effectiveness of the presented methods are out of the scope of this paper.

Our systematic review has limitations typical for that kind of study. We limited articles to those written in English. Papers with keywords beyond our query also could be omitted as well as those not registered in databases we searched through. But, we have done our best to find all relevant papers during the process. Overall, we expect that we have included most of the important studies on the subject.

## VIII. CONCLUSION

The subject of business process execution optimization in services architecture with reliability taken into account is elaborated in many papers. Our review covers publications from 2006 to 2020. We qualified 128 relevant papers for the review. Such a wide scope of papers allows for a broader perspective on the area of application of services to support business processes. We have found papers that describe methods related to optimization stages: Resource Allocation, Service Composition, and Service Scheduling. The vast majority of articles, 119 out of 128 concern Service Composition methods. Only 3 papers were about Service Scheduling, which makes it the least elaborated stage of optimization. The most popular method is a genetic algorithm that is considered the most effective by many researchers. Our review also encompasses papers that describe new original heuristic algorithms relevant to solve problems at the Service Composition stage.

We see the potential for further exploration at both Resource Allocation and Service Scheduling stages. Service composition articles provide inspiration about aspects that could be optimized using service scheduling. Among others, we can name two areas: prioritization according to the business value of the business process, prioritization according to QoS or reliability requirements. The first one is based on the assumption that different business processes give different values to the organization. For example, the sales process may be more important than the warehouse inventory process. In this case, putting more emphasis on the more important process during scheduling may result in their faster execution and higher business value. The second one assumes prioritization based on QoS. In such a model, tasks in the processes with the greatest risk of breaking QoS requirements are performed first. For example, the QoS requirement may mean a limited time to complete the process. For a process executed long enough that there is a risk of breaking the contract, tasks in it have a higher priority in the queue. Such optimization could allow achieving the same QoS requirements with fewer resources.

The problem of optimizing resource allocation is a complex one. We argue that during the deployment phase of the SOA system, usually there are many aspects that should be considered simultaneously. The goal is the optimal realization of business processes, but there are many factors influencing their performance. Typically, there are many servers considered in the allocation, as well as the components running on them. Each component is responsible for a part of the business process implementation. Thus, the size of the resources allocated to the servers is important. The choice of servers and their locations affects the time to transmit data over the network. Also, the deployment of multiple components on the same server has its meaning. More components on one server may reduce data transmission time, but also reduces the number of resources devoted to components deployed. Various processes of varying communication demands may strain network connections and components to a diverse extent. Taking into account all or some of these factors may contribute to better business results. Therefore, we see potential in that area for further exploration.

To sum up, the most popular stage of business process execution optimization in services architecture is Service Composition that considers QoS parameters. Researchers mainly propose heuristic methods that optimize business processes during run-time.

## REFERENCES

[1] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Dept. Comput. Sci., Univ. Durham, Durham, U.K., EBSE Tech. Rep. EBSE-2007-01, 2007.

[2] A. Amjad, F. Azam, M. W. Anwar, W. H. Butt, and M. Rashid, "Event-driven process chain for modeling and verification of business requirements—A systematic literature review," *IEEE Access*, vol. 6, pp. 9027–9048, 2018, doi: 10.1109/ACCESS.2018.2791666.

[3] B. Omair and A. Alturki, "A systematic literature review of fraud detection metrics in business processes," *IEEE Access*, vol. 8, pp. 26893–26903, 2020, doi: 10.1109/ACCESS.2020.2971604.

[4] J. A. Garcia-Garcia, N. Sánchez-Gómez, D. Lizcano, M. J. Escalona, and T. Wojdyński, "Using blockchain to improve collaborative business process management: Systematic literature review," *IEEE Access*, vol. 8, pp. 142312–142336, 2020, doi: 10.1109/ACCESS.2020.3013911.

[5] J. A. Rasheedh and S. Saradha, "Review of micro-services architectures and runtime dynamic binding," in *Proc. 4th Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud) (I-SMAC)*, Palladam, India, Oct. 2020, pp. 1130–1137, doi: 10.1109/I-SMAC49090.2020.9243335.

[6] P. Valderas, V. Torres, and V. Pelechano, "A microservice composition approach based on the choreography of BPMN fragments," *Inf. Softw. Technol.*, vol. 127, Nov. 2020, Art. no. 106370, doi: 10.1016/j.infsof.2020.106370.

[7] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–72, Jul. 1992, doi: 10.1038/scientificamerican0792-66.

[8] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.

[9] C. Zhang, R. N. Chang, C.-S. Perng, E. So, C. Tang, and T. Tao, "An optimal capacity planning algorithm for provisioning cluster-based failure-resilient composite services," in *Proc. IEEE Int. Conf. Services Comput.*, Bengaluru, India, Sep. 2009, pp. 112–119, doi: 10.1109/SCC.2009.81.

[10] L. Xie, J. Luo, J. Qiu, J. A. Pershing, Y. Li, and Y. Chen, "Availability 'weak point' analysis over an SOA deployment framework," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Salvador, Brazil, Apr. 2008, pp. 473–480, doi: 10.1109/NOMS.2008.4575170.

[11] R. Mennes, B. Spinnewyn, S. Latré, and J. F. Botero, "GRECO: A distributed genetic algorithm for reliable application placement in hybrid clouds," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Pisa, Italy, Oct. 2016, pp. 14–20, doi: 10.1109/CloudNet.2016.45.

[12] M. Schmid, "An approach for autonomic performance management in SOA workflows," in *Proc. 12th IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM) Workshops*, Dublin, Ireland, May 2011, pp. 698–701, doi: 10.1109/INM.2011.5990659.

[13] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian, "Resource management in the autonomic service-oriented architecture," in *Proc. IEEE Int. Conf. Autonomic Comput.*, Dublin, Ireland, Jun. 2006, pp. 84–92, doi: 10.1109/ICAC.2006.1662385.

[14] M. Abdullah, W. Iqbal, F. Bukhari, and A. Erradi, "Diminishing returns and deep learning for adaptive CPU resource allocation of containers," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 2052–2063, Dec. 2020, doi: 10.1109/TNSM.2020.3033025.

[15] K.-C. Huang, Y.-C. Lu, M.-H. Tsai, Y.-J. Wu, and H.-Y. Chang, "Performance-efficient service deployment and scheduling methods for composite cloud services," in *Proc. 9th Int. Conf. Utility Cloud Comput.*, Shanghai, China, Dec. 2016, pp. 240–244, doi: 10.1145/2996890.3007866.

[16] P. Czarnul, "Modelling, optimization and execution of workflow applications with data distribution, service selection and budget constraints in BeesyCluster," in *Proc. Int. Multiconference Comput. Sci. Inf. Technol.*, Oct. 2010, pp. 629–636, doi: 10.1109/IMCSIT.2010.5679934.

[17] C. Xiang, W. Zhao, C. Tian, J. Nie, and J. Zhang, "QoS-aware, optimal and automated service composition with Users' constraints," in *Proc. IEEE 8th Int. Conf. e-Business Eng.*, Beijing, China, Oct. 2011, pp. 223–228, doi: 10.1109/ICEBE.2011.59.

[18] Y. Liu, L. Wu, and S. Liu, "A novel QoS-aware service composition approach based on path decomposition," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Guilin, China, Dec. 2012, pp. 76–82, doi: 10.1109/APSCC.2012.26.

[19] Y. Syu, Y.-Y. FanJiang, J.-Y. Kuo, and S.-P. Ma, "Towards a genetic algorithm approach to automating workflow composition for web services with transactional and QoS-awareness," in *Proc. IEEE World Congr. Services*, Washington, DC, USA, Jul. 2011, pp. 295–302, doi: 10.1109/SERVICES.2011.68.

[20] S. A. Ludwig, "Clonal selection based genetic algorithm for workflow service selection," in *Proc. IEEE Congr. Evol. Comput.*, Brisbane, QLD, Australia, Jun. 2012, pp. 1–7, doi: 10.1109/CEC.2012.6256465.

[21] G. A. Ebrahim, "Intelligent composition of dynamic-cost services in service-oriented architectures," in *Proc. UKSim 5th Eur. Symp. Comput. Modeling Simulation*, vol. 1, Nov. 2011, pp. 53–58, doi: 10.1109/EMS.2011.51.

[22] D. Wang, Y. Yang, and Z. Mi, "A genetic-based approach to web service composition in geo-distributed cloud environment," *Comput. Electr. Eng.*, vol. 43, pp. 129–141, Apr. 2015, doi: 10.1016/j.compeleceng.2014.10.008.

[23] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "A framework for QoS-aware binding and re-binding of composite web services," *J. Syst. Softw.*, vol. 81, no. 10, pp. 1754–1769, Oct. 2008, doi: 10.1016/j.jss.2007.12.792.

[24] C. Zhang, S. Su, and J. Chen, "A novel genetic algorithm for QoS-aware web services selection," in *Data Engineering Issues in E-Commerce and Services* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2006, pp. 224–235, doi: 10.1007/11780397_18.

[25] A. S. Da Silva, H. Ma, and M. Zhang, "A graph-based QoS-aware method for web service composition with branching," in *Proc. Genetic Evol. Comput. Conf. Companion*, Denver, CO, USA, Jul. 2016, pp. 131–132, doi: 10.1145/2908961.2909044.

[26] S.-P. Ma, C.-W. Lan, C.-T. Ho, and J.-H. Ye, "QoS-aware selection of web Apis based on $\epsilon$-Pareto genetic algorithm," in *Proc. Int. Comput. Symp. (ICS)*, Chiayi, Taiwan, Dec. 2016, pp. 595–600, doi: 10.1109/ICS.2016.0122.

[27] S. Zhang and I. Paik, "An efficient algorithm for web service selection based on local selection in large scale," in *Proc. IEEE 8th Int. Conf. Awareness Sci. Technol. (iCAST)*, Taichung, Taiwan, Nov. 2017, pp. 188–193, doi: 10.1109/ICAwST.2017.8256443.

[28] W. Radwan, Y. Hassouneh, A. S. Sayyad, and N. Ammar, "YAFA-SOA: A GA-based optimizer for optimizing security and cost in service compositions," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Honolulu, HI, USA, Jun. 2017, pp. 330–337, doi: 10.1109/SCC.2017.49.

[29] A. Klein, F. Ishikawa, and S. Honiden, "SanGA: A self-adaptive network-aware approach to service composition," *IEEE Trans. Services Computing*, vol. 7, no. 3, pp. 452–464, Jul. 2014, doi: 10.1109/TSC.2013.2.

[30] Y. Zhou, C. Zhang, and B. Zhang, "Multi-objective service compositon optimization using differential evolution," in *Proc. 11th Int. Conf. Natural Comput. (ICNC)*, Zhangjiajie, China, Aug. 2015, pp. 233–238, doi: 10.1109/ICNC.2015.7377996.

[31] A. S. da Silva, H. Ma, and M. Zhang, "A GP approach to QoS-aware web service composition including conditional constraints," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Sendai, Japan, May 2015, pp. 2113–2120, doi: 10.1109/CEC.2015.7257145.

[32] M. C. Jaeger and G. Mühl, "QoS-based selection of services: The implementation of a genetic algorithm," in *Proc. Commun. Distrib. Syst. ITG/GI Symp.*, Bern, Switzerland, 2007, pp. 1–12.

[33] Z. Chen, H. Wang, and P. Pan, "An approach to optimal web service composition based on QoS and user preferences," in *Proc. Int. Joint Conf. Artif. Intell.*, Hainan Island, China, Apr. 2009, pp. 96–101, doi: 10.1109/JCAI.2009.206.

[34] H. Wada, P. Champrasert, J. Suzuki, and K. Oba, "Multiobjective optimization of SLA-aware service composition," in *Proc. IEEE Congr. Services*, Honolulu, HI, USA, Jul. 2008, pp. 368–375, doi: 10.1109/SERVICES-1.2008.77.

[35] A. Klein, F. Wagner, F. Ishikawa, and S. Honiden, "A probabilistic approach for long-term B2B service compositions," in *Proc. IEEE 19th Int. Conf. Web Services*, Honolulu, HI, USA, Jun. 2012, pp. 259–266, doi: 10.1109/ICWS.2012.39.

[36] P. P. Beran, E. Vinek, E. Schikuta, and M. Leitner, "An adaptive heuristic approach to service selection problems in dynamic distributed systems," in *Proc. ACM/IEEE 13th Int. Conf. Grid Comput.*, Beijing, China, Sep. 2012, pp. 66–75, doi: 10.1109/Grid.2012.26.

[37] Y.-Y. Fanjiang, Y. Syu, C.-H. Wu, J.-Y. Kuo, and S.-P. Ma, "Genetic algorithm for QoS-aware dynamic web services composition," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Qingdao, China, Jul. 2010, pp. 3246–3251, doi: 10.1109/ICMLC.2010.5580691.

[38] Y. Yu, H. Ma, and M. Zhang, "An adaptive genetic programming approach to QoS-aware web services composition," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, Jun. 2013, pp. 1740–1747, doi: 10.1109/CEC.2013.6557771.

[39] S. Li and Y. Li, "Benefit equilibrium driven selection of web service based on an adaptive genetic algorithm," in *Proc. 9th Int. Conf. Natural Comput. (ICNC)*, Shenyang, China, Jul. 2013, pp. 418–422, doi: 10.1109/ICNC.2013.6818012.

[40] H. Wang, P. Ma, Q. Yu, D. Yang, J. Li, and H. Fei, "Combining quantitative constraints with qualitative preferences for effective non-functional properties-aware service composition," *J. Parallel Distrib. Comput.*, vol. 100, pp. 71–84, Feb. 2017, doi: 10.1016/j.jpdc.2016.10.013.

[41] Y. Zhao, W. Tan, and T. Jin, "QoS-aware web service composition considering the constraints between services," in *Proc. 12th Chin. Conf. Comput. Supported Cooperat. Work Social Comput.*, Chongqing, China, Sep. 2017, pp. 229–232, doi: 10.1145/3127404.3127451.

[42] F. Wagner, F. Ishikawa, and S. Honiden, "Robust service compositions with functional and location diversity," *IEEE Trans. Services Comput.*, vol. 9, no. 2, pp. 277–290, Mar. 2016, doi: 10.1109/TSC.2013.2295791.

[43] Q. Wu, F. Ishikawa, Q. Zhu, and D. H. Shin, "QoS-aware multigranularity service composition: Modeling and optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 11, pp. 1565–1577, Nov. 2016, doi: 10.1109/TSMC.2015.2503384.

[44] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "$E^3$: A multiobjective optimization framework for SLA-aware service composition," *IEEE Trans. Services Comput.*, vol. 5, no. 3, pp. 358–372, Feb. 2012, doi: 10.1109/TSC.2011.6.

[45] C. Buqing, L. Jianxun, X. F. Liu, L. Bing, Z. Dong, and K. Guosheng, "CHC-TSCM: A trustworthy service composition method based on an improved CHC genetic algorithm," *China Commun.*, vol. 10, no. 12, pp. 77–91, Dec. 2013, doi: 10.1109/CC.2013.6723881.

[46] T. Phan and W.-S. Li, "Heuristics-based scheduling of composite web service workloads," in *Proc. 1st Workshop Middleware Service Oriented Comput. (MW SOC)*, Melbourne, VIC, Australia, 2006, pp. 30–35, doi: 10.1145/1169091.1169096.

[47] Y. Vanrompay, P. Rigole, and Y. Berbers, "Genetic algorithm-based optimization of service composition and deployment," in *Proc. 3rd Int. Workshop Services Integr. Pervas. Environ. (SIPE)*, Sorrento, Italy, 2008, pp. 13–18, doi: 10.1145/1387309.1387313.

[48] F. Qiqing, P. Xiaoming, L. Qinghua, and H. Yahui, "A global QoS optimizing web services selection algorithm based on MOACO for dynamic web service composition," in *Proc. Int. Forum Inf. Technol. Appl.*, Chengdu, China, May 2009, pp. 37–42, doi: 10.1109/IFITA.2009.91.

[49] W.-n. Chen, Y. Shi, and J. Zhang, "An ant colony optimization algorithm for the time-varying workflow scheduling problem in grids," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, May 2009, pp. 875–880, doi: 10.1109/CEC.2009.4983037.

[50] W. Zhang, C. K. Chang, T. Feng, and H.-Y. Jiang, "QoS-based dynamic web service composition with ant colony optimization," in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf.*, Seoul, South Korea, Jul. 2010, pp. 493–502, doi: 10.1109/COMPSAC.2010.76.

[51] Z.-J. Wang, Z.-Z. Liu, X.-F. Zhou, and Y.-S. Lou, "An approach for composite web service selection based on DGQoS," *Int. J. Adv. Manuf. Technol.*, vol. 56, nos. 9–12, pp. 1167–1179, Oct. 2011, doi: 10.1007/s00170-011-3230-9.

[52] Z.-Z. Liu, Z.-J. Wang, X.-F. Zhou, Y.-S. Lou, and L. Shang, "A new algorithm for QoS-aware composite web services selection," in *Proc. 2nd Int. Workshop Intell. Syst. Appl.*, Wuhan, China, May 2010, pp. 1–4, doi: 10.1109/IWISA.2010.5473293.

[53] Y. Zhong, W. Li, X. Wang, and J. Fan, "An approach to agent-coalition-based automatic web service composition," in *Proc. IEEE 16th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, Wuhan, China, May 2012, pp. 37–42, doi: 10.1109/CSCWD.2012.6221794.

[54] T. Wang, C. Niu, and L. Cheng, "A two-phase context-sensitive service composition method with the workflow model in cyber-physical systems," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, Chengdu, China, Dec. 2014, pp. 1475–1482, doi: 10.1109/CSE.2014.275.

[55] C. Yu, G. Wang, and Y. Liu, "A multi-agent based architecture for web service selection in E-business," in *Proc. IEEE 8th Int. Conf. e-Business Eng.*, Beijing, China, Oct. 2011, pp. 245–250, doi: 10.1109/ICEBE.2011.19.

[56] D. Li, N. Deng, G. Tan, C. Liu, and J. Chen, "Algorithm for QOS-aware web service composition based on flow path tree with probabilities," in *Proc. 2nd IEEE Int. Conf. Netw. Infrastruct. Digit. Content*, Beijing, China, Sep. 2010, pp. 929–933, doi: 10.1109/ICNIDC.2010.5657934.

[57] J. Liao, Y. Liu, X. Zhu, J. Wang, and Q. Qi, "Accurate QoS-based service selection algorithm for service composition," in *Proc. 38th Annu. IEEE Conf. Local Comput. Netw.*, Sydney, NSW, Australia, Oct. 2013, pp. 344–347, doi: 10.1109/LCN.2013.6761265.

[58] H. Xia and Z. Li, "Particle swarm algorithm for the quality of service-oriented web services selection," in *Proc. 2nd Int. Symp. Knowl. Acquisition Modeling*, Wuhan, China, 2009, pp. 303–306, doi: 10.1109/KAM.2009.187.

[59] Y. Li and S. Li, "Adaptive particle swarm optimization-based web service selection," in *Proc. 9th Int. Conf. Natural Comput. (ICNC)*, Shenyang, China, Jul. 2013, pp. 486–490, doi: 10.1109/ICNC.2013.6818025.

[60] X. Fan, C. Jiang, and X. Fang, "An efficient approach to web service selection," in *Web Information Systems and Mining* (Lecture Notes in Computer Science), vol. 5854, W. Liu, X. Luo, F. L. Wang, and J. Lei, Eds. Berlin, Germany: Springer, 2009, doi: 10.1007/978-3-642-05250-7_29.

[61] M. Mohammed, M. A. Chikh, and H. Fethallah, "QoS-aware web service selection based on harmony search," in *Proc. 4th Int. Symp. ISKO-Maghreb, Concepts Tools Knowl. Manage. (ISKO-Maghreb)*, Algiers, Algeria, Nov. 2014, pp. 1–6, doi: 10.1109/ISKO-Maghreb.2014.7033465.

[62] P. M. Esfahani, J. Habibi, and T. Varaee, "Application of social harmony search algorithm on composite web service selection based on quality attributes," in *Proc. 6th Int. Conf. Genetic Evol. Comput.*, Kitakushu, Japan, Aug. 2012, pp. 526–529, doi: 10.1109/ICGEC.2012.65.

[63] Z. Liu and X. Xu, "S-ABC—A service-oriented artificial bee colony algorithm for global optimal services selection in concurrent requests environment," in *Proc. IEEE Int. Conf. Web Services*, Anchorage, AK, USA, Jun. 2014, pp. 503–509, doi: 10.1109/ICWS.2014.77.

[64] F. Mohr, A. Jungmann, and H. K. Buning, "Automated online service composition," in *Proc. IEEE Int. Conf. Services Comput.*, New York, NY, USA, Jun. 2015, pp. 57–64, doi: 10.1109/SCC.2015.18.

[65] F.-C. Pop, D. Pallez, M. Cremene, A. Tettamanzi, M. Suciu, and M. Vaida, "QoS-based service optimization using differential evolution," in *Proc. 13th Annu. Conf. Genetic Evol. Comput. (GECCO)*, Dublin, Ireland, 2011, pp. 1891–1898, doi: 10.1145/2001576.2001830.

[66] Y. Zhang, G. Cui, Y. Wang, X. Guo, and S. Zhao, "An optimization algorithm for service composition based on an improved FOA," *Tsinghua Sci. Technol.*, vol. 20, no. 1, pp. 90–99, Feb. 2018, doi: 10.1109/TST.2015.7040518.

[67] K. Hashmi, H. AlJafar, Z. Malik, and A. Alhosban, "A bat algorithm based approach of QoS optimization for long term business pattern," in *Proc. 7th Int. Conf. Inf. Commun. Syst. (ICICS)*, Irbid, Jordan, Apr. 2016, pp. 27–32, doi: 10.1109/IACS.2016.7476081.

[68] F. Xiang, G. Z. Jiang, L. Xu, and N. Wang, "The case-library method for service composition and optimal selection of big manufacturing data in cloud manufacturing system," *Int. J. Adv. Manuf. Technol.*, vol. 84, nos. 1–4, pp. 59–70, 2016, doi: 10.1007/s00170-015-7813-8.

[69] W. Nam, H. Kil, and J. Lee, "QoS-driven web service composition using learning-based depth first search," in *Proc. IEEE Conf. Commerce Enterprise Comput.*, Vienna, Austria, Jul. 2009, pp. 507–510, doi: 10.1109/CEC.2009.50.

[70] X. Zhao, Z. Wen, and X. Li, "QoS-aware web service selection with negative selection algorithm," *Knowl. Inf. Syst.*, vol. 40, no. 2, pp. 349–373, Aug. 2014, doi: 10.1007/s10115-013-0642-x.

[71] H. Bannazadeh and A. Leon-Garcia, "Allocating services to applications using Markov decision processes," in *Proc. IEEE Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Newport Beach, CA, USA, Jun. 2007, pp. 141–146, doi: 10.1109/SOCA.2007.7.

[72] I. ul Haq, E. Schikuta, and K. Kofler, "Using blackboard system to automate and optimize workflow orchestrations," in *Proc. Int. Conf. Emerg. Technol.*, Islamabad, Pakistan, Oct. 2009, pp. 173–178, doi: 10.1109/ICET.2009.5353179.

[73] M. Oh, J. Baik, S. Kang, and H.-J. Choi, "An efficient approach for QoS-aware service selection based on a tree-based algorithm," in *Proc. 7th IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Portland, OR, USA, May 2008, pp. 605–610, doi: 10.1109/ICIS.2008.8.

[74] P. P. W. Chan and M. R. Lyu, "Dynamic web service composition: A new approach in building reliable web service," in *Proc. 22nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Gino-Wan, Japan, 2008, pp. 20–25, doi: 10.1109/AINA.2008.133.

[75] M. Li, B. Li, and J. Huai, "Reliability-aware automatic composition approach for web services," *Sci. China Inf. Sci.*, vol. 55, no. 4, pp. 921–937, Apr. 2012, doi: 10.1007/s11432-011-4545-8.

[76] X. Zhang, J. Zhuo, J. Li, and G. Wu, "A learning automation solution to the QoS-aware service composition," in *Proc. Int. Conf. Web Inf. Syst. Mining*, Shanghai, China, Nov. 2009, pp. 297–301, doi: 10.1109/WISM.2009.68.

[77] L. Chen, H. Jian, and J. Wu, "WSCRec: Utilizing historical information to facilitate web service composition," in *Proc. IEEE 19th Int. Conf. Web Services*, Honolulu, HI, USA, Jun. 2012, pp. 633–634, doi: 10.1109/ICWS.2012.121.

[78] X. Zhang and W. Dou, "Preference-aware QoS evaluation for cloud web service composition based on artificial neural networks," in *Web Information Systems and Mining* (Lecture Notes in Computer Science), vol. 6318. Berlin, Germany: Springer, 2010, doi: 10.1007/978-3-642-16515-3_51.

[79] J. Huang, G. Liu, Q. Duan, and Y. Yan, "QoS-aware service composition for converged network-cloud service provisioning," in *Proc. IEEE Int. Conf. Services Comput.*, Anchorage, AK, USA, Jun. 2014, pp. 67–74, doi: 10.1109/SCC.2014.18.

[80] J.-Z. Luo, J.-Y. Zhou, and Z.-A. Wu, "An adaptive algorithm for QoS-aware service composition in grid environments," *Service Oriented Comput. Appl.*, vol. 3, no. 3, pp. 217–226, Sep. 2009, doi: 10.1007/s11761-009-0047-6.

[81] J. A. Parejo, S. Segura, P. Fernandez, and A. Ruiz-Cortés, "QoS-aware web services composition using GRASP with path relinking," *Expert Syst. Appl.*, vol. 41, no. 9, pp. 4211–4223, Jul. 2014, doi: 10.1016/j.eswa.2013.12.036.

[82] B. Wu, C.-H. Chi, Z. Chen, M. Gu, and J. Sun, "Workflow-based resource allocation to optimize overall performance of composite services," *Future Gener. Comput. Syst.*, vol. 25, no. 3, pp. 199–212, Mar. 2009, doi: 10.1016/j.future.2008.06.003.

[83] D. A. Menascé, E. Casalicchio, and V. Dubey, "On optimal service selection in service oriented architectures," *Perform. Eval.*, vol. 67, no. 8, pp. 659–675, Aug. 2010, doi: 10.1016/j.peva.2009.07.001.

[84] D. A. Menascé, E. Casalicchio, and V. Dubey, "A heuristic approach to optimal service selection in service oriented architectures," in *Proc. 7th Int. Workshop Softw. Perform. (WOSP)*, Princeton, NJ, USA, 2008, pp. 13–24, doi: 10.1145/1383559.1383562.

[85] W. Tan, Y. Sun, L. X. Li, G. Lu, and T. Wang, "A trust service-oriented scheduling model for workflow applications in cloud computing," *IEEE Syst. J.*, vol. 8, no. 3, pp. 868–878, Sep. 2014, doi: 10.1109/JSYST.2013.2260072.

[86] M. Bakhshi, F. Mardukhi, and N. Nematbakhsh, "A fuzzy-based approach for selecting the optimal composition of services according to user preferences," in *Proc. 2nd Int. Conf. Comput. Autom. Eng. (ICCAE)*, Singapore, Feb. 2010, pp. 129–135, doi: 10.1109/ICCAE.2010.5451983.

[87] Z. Wang, F. Xu, and X. Xu, "A cost-effective service composition method for mass customized QoS requirements," in *Proc. IEEE 9th Int. Conf. Services Comput.*, Honolulu, HI, USA, Jun. 2012, pp. 194–201, doi: 10.1109/SCC.2012.5.

[88] G. Zhang and Z. Wang, "A QoS-aware service composition optimization based on logical structure," in *Proc. Int. Conf. Comput. Intell. Softw. Eng.*, Wuhan, China, Dec. 2009, pp. 1–5, doi: 10.1109/CISE.2009.5365231.

[89] S.-L. Pan and Q.-J. Mao, "Semantic web service composition planner agent with a QoS-aware selection model," in *Proc. Int. Conf. Web Inf. Syst. Mining*, Shanghai, China, Nov. 2009, pp. 325–331, doi: 10.1109/WISM.2009.74.

[90] Y.-S. Luo, Y. Qi, L.-F. Shen, D. Hou, C. Sapa, and Y. Chen, "An improved heuristic for QoS-aware service composition framework," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun.*, Dalian, China, Sep. 2008, pp. 360–367, doi: 10.1109/HPCC.2008.17.

[91] C.-M. Wang, S.-T. Wang, H.-M. Chen, and C.-C. Huang, "A reliability-aware approach for web services execution planning," in *Proc. IEEE Congr. Services (Services)*, Salt Lake City, UT, USA, Jul. 2007, pp. 278–283, doi: 10.1109/SERVICES.2007.9.

[92] C. Surianarayanan, G. Ganapathy, and M. S. Ramasamy, "An approach for selecting best available services through a new method of decomposing QoS constraints," *Service Oriented Comput. Appl.*, vol. 9, pp. 107–138, 2014, doi: 10.1007/s11761-014-0154-x.

[93] B. Sahoo and P. Bhuyan, "A selection approach in service composition of SOA," in *Proc. Int. Conf. Recent Trends Inf. Technol. (ICRTIT)*, Chennai, India, Apr. 2016, pp. 1–6, doi: 10.1109/ICRTIT.2016.7569573.

[94] J. Eckert, D. Ertogrul, A. Miede, N. Repp, and R. Steinmetz, "Resource planning heuristics for service-oriented workflows," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Sydney, NSW, Australia, Dec. 2008, pp. 591–597, doi: 10.1109/WIIAT.2008.108.

[95] S. Y. Hwang, E. P. Lim, C. H. Lee, and C. H. Chen, "Dynamic web service selection for reliable web service composition," *IEEE Trans. Services Computing*, vol. 1, no. 2, pp. 104–116, Apr. 2008, doi: 10.1109/TSC.2008.2.

[96] L. Xi, Q. Zhou, and L. Xie, "An approach on service component selection and analysis of the resulting system reliability," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol.*, Bradford, U.K., Jun. 2010, pp. 1246–1250, doi: 10.1109/CIT.2010.225.

[97] I. Guidara, N. Guermouche, T. Chaari, S. Tazi, and M. Jmaiel, "Heuristic based time-aware service selection approach," in *Proc. IEEE Int. Conf. Web Services*, New York, NY, USA, Jun. 2015, pp. 65–72, doi: 10.1109/ICWS.2015.19.

[98] X. Deng and C. Xing, "A QoS-oriented optimization model for web service group," in *Proc. 8th IEEE/ACIS Int. Conf. Comput. Inf. Sci.*, Shanghai, China, Jun. 2009, pp. 903–909, doi: 10.1109/ICIS.2009.91.

[99] Y. Xia, P. Chen, L. Bao, M. Wang, and J. Yang, "A QoS-aware web service selection algorithm based on clustering," in *Proc. IEEE Int. Conf. Web Services*, Washington, DC, USA, Jul. 2011, pp. 428–435, doi: 10.1109/ICWS.2011.36.

[100] D. Liu, Z. Shao, C. Yu, and G. Fan, "A heuristic QoS-aware service selection approach to web service composition," in *Proc. 8th IEEE/ACIS Int. Conf. Comput. Inf. Sci.*, Shanghai, China, Jun. 2009, pp. 1184–1189, doi: 10.1109/ICIS.2009.76.

[101] M. Li, D. Zhu, T. Deng, H. Sun, H. Guo, and X. Liu, "GOS: A global optimal selection strategies for QoS-aware web services composition," *Service Oriented Comput. Appl.*, vol. 7, no. 3, pp. 181–197, Sep. 2013, doi: 10.1007/s11761-013-0133-7.

[102] D. Schuller, J. Eckert, A. Miede, S. Schulte, and R. Steinmetz, "QoS-aware service composition for complex workflows," in *Proc. 5th Int. Conf. Internet Web Appl. Services*, Barcelona, Spain, 2010, pp. 333–338, doi: 10.1109/ICIW.2010.55.

[103] M. Živkovic, J. W. Bosman, H. van den Berg, R. van der Mei, H. B. Meeuwissen, and R. Nunez-Queija, "Run-time revenue maximization for composite web services with response time commitments," in *Proc. IEEE 26th Int. Conf. Adv. Inf. Netw. Appl.*, Fukuoka, Japan, Mar. 2012, pp. 589–596, doi: 10.1109/AINA.2012.25.

[104] J. Wang, J. Wu, Y. Wu, and S. Zhang, "MARDO: A novel schema of dynamic QoS optimizing for composite web services using cooperative agents," in *Proc. 14th Int. Conf. Comput. Supported Cooperat. Work Design*, Shanghai, China, Apr. 2010, pp. 408–412, doi: 10.1109/CSCWD.2010.5471940.

[105] T. Ahmed and A. Srivastava, "Minimizing waiting time for service composition: A frictional approach," in *Proc. IEEE 20th Int. Conf. Web Services*, Santa Clara, CA, USA, Jun. 2013, pp. 268–275, doi: 10.1109/ICWS.2013.44.

[106] M. Li, T. Deng, H. Sun, H. Guo, and X. Liu, "GOS: A global optimal selection approach for QoS-aware web services composition," in *Proc. 5th IEEE Int. Symp. Service Oriented Syst. Eng.*, Nanjing, China, Jun. 2010, pp. 7–14, doi: 10.1109/SOSE.2010.47.

[107] J. Jiang and Q. Bai, "Correlated contribution analysis for service composition in dynamic environments," in *Proc. IEEE Int. Conf. Services Comput.*, Santa Clara, CA, USA, Jun. 2013, pp. 113–119, doi: 10.1109/SCC.2013.85.

[108] D. Lee, H. Shin, and E. Park, "Modeling recovery strategies in service-oriented architecture using a Markov decision process," in *Proc. IEEE 13th Int. Symp. High-Assurance Syst. Eng.*, Boca Raton, FL, USA, Nov. 2011, pp. 285–290, doi: 10.1109/HASE.2011.25.

[109] S. Kalasapur, M. Kumar, and B. A. Shirazi, "Dynamic service composition in pervasive computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 7, pp. 907–918, Jul. 2007, doi: 10.1109/TPDS.2007.1039.

[110] R. Ukor and A. Carpenter, "Goal-oriented service selection in business processes," in *Proc. 4th Int. Conf. Softw. Eng. Adv.*, Porto, Portugal, Sep. 2009, pp. 484–489, doi: 10.1109/ICSEA.2009.76.

[111] F. Tao, Y. J. LaiLi, L. D. Xu, and L. Zhang, "FC-PACO-RM: A parallel method for service composition optimal-selection in cloud manufacturing system," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2023–2033, Nov. 2013, doi: 10.1109/TII.2012.2232936.

[112] E. Oppong and S. Khaddaj, "SOA and cloud service provisioning framework," in *Proc. 11th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, Guilin, China, Oct. 2012, pp. 200–204, doi: 10.1109/DCABES.2012.104.

[113] W. Fdhila, M. Dumas, and C. Godart, "Optimized decentralization of composite web services," in *Proc. 6th Int. Conf. Collaborative Comput., Netw., Appl., Worksharing (ICST)*, Chicago, IL, USA, 2010, pp. 1–10, doi: 10.4108/icst.collaboratecom.2010.1.

[114] M. L. M. Peixoto, M. J. Santana, and R. H. C. Santana, "A P2P hierarchical metascheduler to obtain QoS in a grid economy services," in *Proc. Int. Conf. Comput. Sci. Eng.*, Vancouver, BC, Canada, 2009, pp. 292–297, doi: 10.1109/CSE.2009.385.

[115] J. Bian, C. Weng, J. Du, and M. Li, "A QoS-aware and fault-tolerant workflow composition for grid," in *Proc. 7th Int. Conf. Grid Cooperat. Comput.*, Shenzhen, China, Oct. 2008, pp. 510–516, doi: 10.1109/GCC.2008.95.

[116] P. Kathiravelu, T. G. Grbac, and L. Veiga, "A FIRM approach for software-defined service composition," in *Proc. 39th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, Opatija, Croatia, May 2016, pp. 565–570, doi: 10.1109/MIPRO.2016.7522206.

[117] J. Andersson, A. Heberle, J. Kirchner, and W. Lowe, "Service level achievements—Distributed knowledge for optimal service selection," in *Proc. IEEE 9th Eur. Conf. Web Services*, Lugano, Switzerland, Sep. 2011, pp. 125–132, doi: 10.1109/ECOWS.2011.24.

[118] I. Guidara, T. Chaari, and M. Jmaiel, "An efficient service selection approach with time-dependent QoS," in *Proc. IEEE 23rd Int. WETICE Conf.*, Parma, Italy, Jun. 2014, pp. 320–325, doi: 10.1109/WETICE.2014.10.

[119] R. Ukor and A. Carpenter, "Flexible service selection optimization using meta-metrics," in *Proc. Congr. Services-I*, Los Angeles, CA, USA, Jul. 2009, pp. 593–598, doi: 10.1109/SERVICES-I.2009.71.

[120] R. Ramacher and L. Monch, "Reliable service reconfiguration for time-critical service compositions," in *Proc. IEEE Int. Conf. Services Comput.*, Santa Clara, CA, USA, Jun. 2013, pp. 184–191, doi: 10.1109/SCC.2013.44.

[121] W. Dong and L. Jiao, "QoS-aware web service composition based on SLA," in *Proc. 4th Int. Conf. Natural Comput.*, Jinan, China, 2008, pp. 247–251, doi: 10.1109/ICNC.2008.705.

[122] J. E. Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz, "QoS-driven selection of web services for transactional composition," in *Proc. IEEE Int. Conf. Web Services*, Beijing, China, Sep. 2008, pp. 653–660, doi: 10.1109/ICWS.2008.116.

[123] D. Dyachuk and R. Deters, "Service level agreement aware workflow scheduling," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Salt Lake City, UT, USA, Jul. 2007, pp. 715–716, doi: 10.1109/SCC.2007.99.

[124] D. Dyachuk and R. Deters, "Ensuring service level agreements for service workflows," in *Proc. IEEE Int. Conf. Services Comput.*, Honolulu, HI, USA, Jul. 2008, pp. 333–340, doi: 10.1109/SCC.2008.117.

[125] H. Alayed, F. Dahan, T. Alfakih, H. Mathkour, and M. Arafah, "Enhancement of ant colony optimization for QoS-aware web service selection," *IEEE Access*, vol. 7, pp. 97041–97051, 2019, doi: 10.1109/ACCESS.2019.2927769.

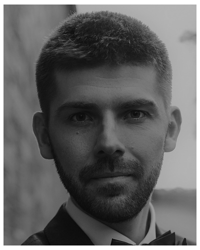
[126] A. Palii and R. Novogrudska, "Web-services selection based on fuzzy preference relations," in *Proc. Int. Conf. Inf. Telecommun. Technol. Radio Electron. (UkrMiCo)*, Odessa, Ukraine, Sep. 2017, pp. 1–4, doi: 10.1109/UkrMiCo.2017.8095431.

[127] R. Takahashi, K. Nishida, and Y. Fukazawa, "Recommendation method for service selection algorithm based on user preference," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng. (CSAE)*, 2018, pp. 1–6, doi: 10.1145/3207677.3277946.

[128] C. Wang, H. Ma, G. Chen, S. Hartmann, and J. Branke, "Robustness estimation and optimisation for semantic web service composition with stochastic service failures," *IEEE Trans. Emerg. Topics Comput. Intell.*, early access, Oct. 13, 2020, doi: 10.1109/TETCI.2020.3027870.

[129] N. E. Allali, M. Fariss, H. Asaidi, and M. Bellouki, "Semantic web services composition model using ant colony optimization," in *Proc. 4th Int. Conf. Intell. Comput. Data Sci. (ICDS)*, Fez, Morocco, Oct. 2020, pp. 1–5, doi: 10.1109/ICDS50568.2020.9268756.
[130] L. Zhao, W. Tan, N. Xie, and L. Huang, "An optimal service selection approach for service-oriented business collaboration using crowd-based cooperative computing," *Appl. Soft Comput.*, vol. 92, Jul. 2020, Art. no. 106270, doi: 10.1016/j.asoc.2020.106270.
[131] W. Viriyasitavat and Z. Bi, "Service selection and workflow composition in modern business processes," *J. Ind. Inf. Integr.*, vol. 17, Mar. 2020, Art. no. 100126, doi: 10.1016/j.jii.2020.100126.
[132] C. Wang, H. Ma, A. Chen, and S. Hartmann, "GP-based approach to comprehensive quality-aware automated semantic web service composition," in *Simulated Evolution and Learning* (Lecture Notes in Computer Science), vol. 10593, Y. Shi, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-68759-9_15.
[133] F. Dahan, K. E. Hindi, and A. Ghoneim, "Enhanced artificial bee colony algorithm for QoS-aware web service selection problem," *Computing*, vol. 99, no. 5, pp. 507–517, 2017, doi: 10.1007/s00607-017-0547-8.
[134] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *J. Intell. Manuf.*, vol. 29 no. 8, pp. 1773–1792, Dec. 2018, doi: 10.1007/s10845-016-1215-0.
[135] S. K. Gavvala, C. Jatoth, G. R. Gangadharan, and R. Buyya, "QoS-aware cloud service composition using eagle strategy," *Future Gener. Comput. Syst.*, vol. 90, pp. 273–290, Jan. 2019, doi: 10.1016/j.future.2018.07.062.
[136] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti, "CSA-WSC: Cuckoo search algorithm for web service composition in cloud environments," *Soft Comput.*, vol. 22, no. 24, pp. 8353–8378, 2017, doi: 10.1007/s00500-017-2783-4.

**TOMASZ GÓRSKI** (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the Military University of Technology (MUT), Warsaw, Poland, in 1997 and 2000, respectively. For ten years, he served at the Computer Science Center of the General Staff of the Polish Armed Forces and ended his military service as a major. After getting the Ph.D. degree, he worked on many commercial software development projects, such as check-in system for the Polish Border Guard, financial systems integration for Zone Vision Ltd., London. From 2004 to 2017, he worked as a Professor Assistant at the MUT and built the Center for Advanced Studies in Systems Engineering. From 2005 to 2020, he ran the consulting company RightSolution, an IBM Authorized Training Provider for the rational brand. He has been an IBM rational certified instructor of rational unified process, requirements management, object-oriented analysis and design, and Java programming. He currently works as a Professor Assistant and the Head of the IT Systems Department at the Polish Naval Academy, Gdynia. He teaches object-oriented programming. He is the author of the Architectural views model 1+5, UML Profile and Diagram for Integration Flows, Use Case API Design Pattern, UML Profile for Distributed Ledger Deployment, and Smart Contract Design Pattern. His research interests include software engineering, model-driven engineering, and blockchain. He is a member of the IEEE Computer Society. He shares his experience as an Author and a Reviewer of IEEE Access journal, a member of the International Conference on Systems Engineering Program Committee, and an Editorial Board Member of the *Open Computer Science Journal*.



**ADRIAN P. WOŹNIAK** received the degree in information technology (IT) from the Military University of Technology, Warsaw, in 2012. Subsequently, he worked as a system and business analyst. In this role, he designed many SOA systems. After that, his position changed to an architect and then the main IT architect at the largest retail company in Poland, which allowed him to gain extensive experience in the field of the SOA systems. He is currently a Lecturer in software engineering, system design, and system integration with the Military University of Technology. The subjects he teaches are also his main areas of research.