Applied Artificial Intelligence

An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

A Multiobjective Discrete Grey Wolf Optimization Approach for Transactional and QoS-driven Web Services Composition

Sunita Jalal & Dharmendra Kumar Yadav

To cite this article: Sunita Jalal & Dharmendra Kumar Yadav (2021) A Multiobjective Discrete Grey Wolf Optimization Approach for Transactional and QoS-driven Web Services Composition, Applied Artificial Intelligence, 35:15, 1646-1684, DOI: 10.1080/08839514.2021.1989168

To link to this article: https://doi.org/10.1080/08839514.2021.1989168

Published online: 09 Nov 2021.

APPLIED ARTIFICIAL INTELLIGENCE
2021, VOL. 35, NO. 15, 1646–1684
https://doi.org/10.1080/08839514.2021.1989168

A Multiobjective Discrete Grey Wolf Optimization Approach for Transactional and QoS-driven Web Services Composition

Sunita Jalal[a] and Dharmendra Kumar Yadav[b]

[a]Department of Computer Engineering, College of Technology, Pantnagar, Uttarakhand, India;
[b]Department of Computer Science and Engineering, MNNIT Allahabad, Prayagraj, Uttar Pradesh, India

ABSTRACT
Web services facilitate reusability that allows cost-effective development of business applications using web services composition. Due to the proliferation of web services, different service providers are providing similar functionality web services. But these web services can have different values for QoS attributes and transactional properties. Thus, it is difficult to build a transactional and QoS optimal composite web service. Most of the existing works used the scalarization-based method for selecting optimal composite web service. In the scalarization-based method, the service user should have a priori knowledge of its preferences about the nonfunctional requirements of desired solutions. This paper proposes a Multiobjective Discrete Grey Wolf Optimization (MDGWO)-based approach for Transactional and QoS-driven Web Services Composition. The Pareto dominance concept is used to select optimal composite web service. Generational Distance (GD), Inverse Generational Distance (IGD), and Spread measures are used to evaluate the performance of the proposed approach. Experimental results indicate that the proposed approach performs well.

ARTICLE HISTORY
Compiled 19 October 2021

Introduction

Web services are self-descriptive, loosely coupled, and platform-independent software components that can be published and accessed over a network using open standards (Alonso et al. 2004). Service providers publish descriptions of their services on the service registry. The service user sends a query to the service registry, which discovers services on request through a query. A web service is developed to perform a specific function such as Flight Booking Service, Payment Service, etc. Due to the adoption of cloud computing and service-oriented architecture, web services are playing a significant role in software development. Organizations use web services composition to develop business applications cost-effectively. Web services composition promotes code reusability since the developer has no need to rewrite the code for a

CONTACT Sunita Jalal sunita.jalal@gmail.com Department of Computer Engineering, College of Technology, Gbpuat, Pantnagar, Uttarakhand, India

function from scratch. Web services composition combines and coordinates existing value-added web services with different functionalities to develop a composite web service. A composite web service represents a complete business application that consists of different activities. Web services are building blocks for business application as they perform these activities that can involve long-running transactions. An example of composite service is Online Trip Management Service, which can be developed by aggregating web services for hotel booking, flight booking, and payment.

Design time specification of a composite web service is depicted by a workflow model that consists of a set of abstract component services interconnected using different workflow patterns. The description of different workflow patterns is explained in Van der Aalst et al. (2003). A web service discovery approach is used to find a set of candidate web services for each abstract component service based on its functionality. Figure1 shows an example of a workflow model of composite service CS. The workflow in Figure 1 consists of six abstract component services that are interconnected using workflow patterns. Suppose that each abstract component service has n number of candidate web services to perform its function. Then, the number of possible web services compositions for composite service CS is n^6.

The web services composition can have two types of requirements: Quality of Service (QoS) requirements and Transactional requirements (Abbassi et al. 2015). Quality of Service (QoS) attributes of a web service describes how the service carries out its function. QoS has a significant impact on web service selection to meet the QoS needs of the service requester. Some of the quality of service (QoS) attributes of web services are response time, cost, throughput, availability, and reliability. The lower value of some QoS attributes, such as cost and response time, indicates good quality. For some QoS attributes, such as throughput and reliability, a higher value indicates good quality. Specifications of QoS requirements are defined in Service Level Agreement (SLA), a contract between the service requester and the service provider (Statovci- Halimi and Halimi, 2004). An example of QoS requirement is given as follows: the reliability value of the composite service CS shall be greater than 95%. SLA assures the service requester to get the service as per the specification. Failing to meet specifications defined in SLA could create severe consequences for a service provider. Transactional requirements emphasize correct execution of the service to get a consistent outcome. The application designer defines transactional requirements as a set of transactional properties of the composite service. A transactional property of a web service specifies its behavior in the case of failure. For example, the effect of service having a compensatable transaction property can be compensated after its successful execution. A composite service instance should satisfy the specified QoS and transactional requirements.

With the wider acceptance of cloud computing, IOT, and web-based technologies, more and more functionally equivalent web services having different QoS attributes and transactional properties are available in the service registry. Thus, many composition solutions are possible for a user request with varying values of QoS and transactional properties. Finding a composition solution that satisfies QoS requirements specified in SLA is the NP-hard multiobjective optimization problem (Wada et al. 2011). Thus, finding an optimal composition solution that satisfies both quality of service constraints defined in SLA and transactional requirements defined by the composition designer is also the NP-hard multiobjective optimization problem.

Metaheuristic algorithms can find a near-optimal solution in a reasonable time. Many researchers have focused separately on the web services composition either from the QoS aspect (Canfora et al. 2005; Da Silva, Ma, and Zhang 2016; Karimi, Isazadeh, and Rahmani 2017; Wang, Huang, and Xie 2014; Yao and Chen 2009) or from the transactional aspect (Bhiri, Perrin, and Godart 2005, 2006; Li, Liu, and Wang 2007; Liu et al. 2009). Some researchers have also studied both QoS and transactional aspects in the web services composition (Ding et al. 2015; El Hadad, Manouvrier, and Rukoz 2010; Graiet et al. 2016; Imed and Graiet 2017; Wu and Zhu 2013). Most of the existing works solve the transaction and QoS-based multiobjective web services composition problem using the scalarization-based method by converting it into a single objective problem. They considered the weighted average of all objectives (QoS attributes). With this approach, it is difficult to give proper weights to all QoS attributes when they are conflicting. It produces only one optimal solution that may be nondominated. It is not useful when a service composer requires nondominated solutions to take decision according to the importance of each QoS attribute.

By considering the drawback of the scalarization-based method, this paper proposes a Multiobjective Discrete Grey Wolf Optimization (MDGWO)-based approach for transactional and QoS-driven web services composition. The contributions of the proposed approach can be presented as follows:

(1) Derivation rules are described for determining the transactional properties of the composition of two web services under different workflow patterns.

(2) A tree representation of composition workflow is discussed to compute QoS values of web services composition and to determine transactional properties of composite service using automaton.

(3) The Multiobjective Discrete Grey Wolf Optimization (MDGWO) approach is explained to find nondominated solutions for transactional and QoS-driven web services composition problems.

The rest of this paper is organized as follows. Section 2 describes QoS computation and transactional properties of composite service and defines the problem statement. Section 3 reviews the related work. Section 4 discusses the proposed composition approach based on the tree model of workflow and multiobjective discrete gray wolf optimization. Section 5 demonstrates experimental results, and section 6 presents the conclusion of this paper.

Composite Service Description and Problem Statement

In this section, workflow patterns of composite web service are described and QoS attributes and transactional properties of composite service are explored. Then, the problem statement is specified.

Workflow Patterns in Web Services Composition

Sequential, parallel, conditional, and loop are the four most commonly used workflow patterns considered in web services composition (Cremene et al. 2016). In the sequential pattern ($\rightarrow$), services are executed in sequential order. In the parallel pattern (+), services are executed independently and the next service will not start execution until all parallel services have finished their execution. In the conditional or choice pattern (x), one of the services is executed based on the condition. The service within a loop pattern (*) is executed repeatedly, provided that the loop condition is met. In the workflow model shown in Figure 1, service S2 and service S3 are parallel after successfully executing service S1. Service S4 will start execution after the completion of both S2 and S3. After successful execution of S4, either service S5 or service S6 is executed according to the condition. Suppose that the workflow model shown in Figure 1 represents Online Trip Management Service (OTMS). Service S1 is the Customer Request Service (CRS) that receives details for a trip from the customer. Services S2 and S3 are the Hotel Booking Service (HBS) and Flight Booking Service (FBS), respectively. Service S4 is a Payment Service (PS) that processes payment of the trip. Services S5 and S6 are the Trip Document E-Mail Service (TDES) and Trip Document Message Service (TDMS), respectively. Business Process Execution Language (BPEL) (Jordan et al. 2007) is an example of composition languages. It composes different web services using sequential, choice, parallel, and loop workflow patterns.

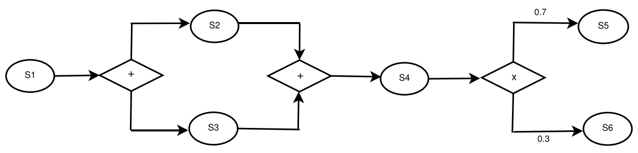


Figure 1. Workflow model.

Quality of Service (QoS) Computation of Composition

Vector Q = {q_1, q_2, …, q_m} denotes the different QoS attributes, where qi represents the ith QoS attribute and m is the total number of considered QoS attributes. QoS attributes can be categorized into two groups: Q_{high} and Q_{low}. For Q_{high}, a larger value of an attribute indicates better quality, such as reliability and availability. But for Q_{low}, a lower value of an attribute indicates better quality, such as response time and service cost. Some of the quality of service attributes are response time, cost, relaibility, and availability. The aggregated value of each QoS attribute of a composite web service depends on the workflow pattern of composition and QoS attribute value of each component web service. Table 1 shows the computation of four QoS attributes under sequential, parallel, choice, and loop workflow patterns.

In the sequential composition pattern, response time and cost are additive, while reliability and availability of composition are multiplicative. In the parallel composition, the cost of composition is additive, while reliability and availability of composition are multiplicative. The response time of parallel composition is the highest response time among all component services. In the choice composition pattern, n component services S1, S2,., Sn are invoked with probabilities p_1, p_2, …, p_n such that $\sum_{i=1}^{n} p_i = 1$.

Transactional Properties of Composition

Web services involved in the composition can fail due to several reasons like machine failure, dynamic changes in the execution environment, and networking issues. Transaction support is required to ensure reliability in web service composition. The main transactional properties of a web service are pivot (p), retriable (r), and compensatable (c) (Mehrotra et al. 1992). A web service can have a combination of transactional properties such as compensatable and retriable (cr) and pivot and retriable (pr). The complete set of transactional properties for a web service is {p, r, c, cr, pr}. The description of properties is given as follows:

• **Retriable Service**: A web service is said to be retriable (r) if it is guaranteed to be complete successfully after a finite number of invocations.

• **Compensatable Service**: A web service is said to be compensatable (c) if it is able to provide compensation policies to undo its effect semantically.

Table 1. Quality of Service (QoS) computation.

QoS attribute	Sequential	Parallel	Choice	Loop
Response time (T)	$\sum_{i=1}^{n} T_i$	$\max_{i\in(1,n)} \{T_i\}$	$\sum_{i=1}^{n} p_i.T_i$	k.T
Cost (C)	$\sum_{i=1}^{n} C_i$	$\sum_{i=1}^{n} C_i$	$\sum_{i=1}^{n} p_i.C_i$	k.C
Availability (A)	$\prod_{i=1}^{n} A_i$	$\prod_{i=1}^{n} A_i$	$\sum_{i=1}^{n} p_i.A_i$	A^k
Reliability (R)	$\prod_{i=1}^{n} R_i$	$\prod_{i=1}^{n} R_i$	$\sum_{i=1}^{n} p_i.R_i$	R^k

• **Pivot Service**: A web service has the pivot (p) transactional property if it is neither retriable nor compensatable. It means that once the pivot web service performs successfully, we cannot semantically undo its effects. Failure of a pivot web service makes no effect at all.
Compensatable and retriable services support backward recovery (i.e., undo) and forward recovery (i.e., redo), respectively. Component web services supporting backward and forward recoveries are useful in maintaining composite web services as failure-atomic. The partial order among transactional properties is as follows: $p < c$ and $pr < r < cr$

Atomic composite web service: To ensure that composite web service is failure-atomic(a), the effects of previously successfully executed component web services have to be semantically undone in the case of unsuccessful execution of one of its component web services. Once a failure-atomic composite web service executes successfully, its effect remains everlasting. The nonatomic transactional property is denoted by (a'). The composite service will be nonatomic (a') for every composition workflow pattern if even one of its component services is nonatomic (a').

A transactional score (t_score) is assigned to each transactional property according to partial order. We assign a t_score value of 1 for either cr web service or failure-atomic (a) web service. Both compensatable and retriable web services are desirable in web services composition, and pivot web service is least desirable. Thus, a t_score value of 0.25 is assigned for pivot web service. Both the compensatable web service and the retriable web service are assigned a t_score value of 0.75. Pivot and retriable(pr) web services are assigned a t_score value of 0.5.

The transactional property of the composite web service is derived by considering workflow patterns and the transactional properties of component web services. El Hadad, Manouvrier, and Rukoz (2010) describe the derivation of transactional properties of composite service under sequential and parallel workflow patterns. Wu and Zhu (2013) also define the derivation of transactional properties of composite service under choice and loop workflow patterns. In a sequential pattern, services are processed from left to right to derive the transactional property of the composition. In parallel and choice patterns, services are processed from top to bottom to derive the transactional property of composition. There must be a guarantee of failure atomicity during the execution of composite web service.

Composition of two component services: The derivation of transactional properties for composite web service under sequential, parallel, choice, and loop workflow patterns is illustrated in Table 2. We are considering only two component services at a time in a workflow pattern.

In a sequential workflow pattern, if the transactional property assigned to the first component web service is p and the second component web service is c, then the transactional property of the composite web service will be nonatomic (a'). In this case, if the first component service executes successfully and

Table 2. Transactional property of composition of two component services.

Web service 1	Web service 2	Sequential ($\rightarrow$)	Parallel (+)	Choice (x)
p	p	a′	a′	p
p	c	a′	a′	p
p	r	p	a′	p
p	pr	a	a′	p
p	cr	a	a	p
c	p	a	a′	p
c	c	c	c	c
c	r	a	a′	p
c	pr	a	a′	p
c	cr	c	c	c
r	p	a′	a′	p
r	c	a′	a′	a′
r	r	r	r	r
r	pr	r	r	r
r	cr	r	r	r
pr	p	a′	a′	p
pr	c	a′	a′	p
pr	r	r	r	r
pr	pr	pr	pr	pr
pr	cr	a	a	pr
cr	p	a	a	p
cr	c	c	c	c
cr	r	r	r	r
cr	pr	pr	pr	pr
cr	cr	cr	cr	cr

the second component service fails, then effects of the first component service cannot be semantically undone. If the transactional property of the first component service is c and the transactional property of the second component service is from set {p, r, pr}, then the composite service will be failure-atomic. If the second component service fails, then the effects of the first component service can be semantically undone.

In a parallel workflow pattern, the composite web service will be atomic if it is created by the parallel composition of p web service and cr web service. If cr service fails after successful completion of p service, then cr service can be retried to complete successfully. The composite web service created by the parallel composition of r web service and p web service will be nonatomic because the effects of r service cannot be undone on the failure of p service. In the choice composition pattern, only one of the web services is executed. For example, choice composition of c web service and p web service will be pivot (p) because if first service fails, then its effects can be compensated and second service will execute to perform the task. If pr web service and r web service are composed using a choice pattern, then the resulting composite service will be r.

Composition of component service and atomic service: Transactional properties of the composition of two services are shown in Table 3, where one service is a component web service and the other is a composite web service. The sequential composition of pivot (p) component web service and atomic (a)

Table 3. Transactional property of composition of component service and composite service.

Web service	Composite Web service	Sequential (→)	Parallel (+)	Choice (x)
p	a	a′	a′	a
c	a	a	a′	a
r	a	a′	a′	a
pr	a	a′	a′	a
cr	a	a	a	a

composite web service will be nonatomic. If the pivot (p) service completes and the atomic (a) service fails, then the effects of the pivot (p) service cannot be semantically undone. The sequential composition of compensatable (c) component web service and atomic (a) composite web service will be atomic. The effects of compensatable service can be semantically undone in case atomic service fails. The parallel composition of compensatable and retriable (cr) component web service and atomic (a) composite web service will be atomic. If the cr service fails, then it can be retried for its successful completion. If cr service completes and atomic service fails, then the effects of cr service can be semantically undone. The choice composition between any web service and atomic composite service will result in atomic composite service.

Composition of two composite services: Transactional properties of the composition of two composite services are presented in Table 4. The sequential composition of two atomic composite web services will be nonatomic. If the first atomic service completes and the second atomic service fails, then the effects of first atomic service cannot be semantically undone. The sequential composition of atomic (a) service and retriable (r) service will be atomic. If atomic service completes and retriable service fails, then retriable service can be re-executed for its successful completion. The parallel composition of atomic (a) service and retriable (r) service will be nonatomic. If atomic service fails, then the effects of retriable service cannot be semantically undone.

Table 4. Transactional property of composition of two composite services.

Composite Web service 1	Composite Web service 2	Sequential (→)	Parallel (+)	Choice (x)
a	p	a′	a′	a
a	c	a′	a′	a
a	r	a	a′	a
a	pr	a	a′	a
a	cr	a	a	a
a	a	a′	a′	a

Table 5. Transactional property of composition under the loop pattern.

Web service	p	c	r	pr	cr
loop(*)	a′	c	r	a′	cr

Composition under the loop pattern: A web service in the loop pattern is executed several times in sequence. If a web service with the transactional property from the set {c, r, cr}, is executed in a loop pattern, its final transactional property will be the same as initial. However, execution of a pivot (p) web service in the loop pattern will be nonatomic. The transactional property under the loop pattern is given in Table 5.

The above rules of transactional properties can be used to obtain the transactional property of a composite web service that contains a number of component web services in complex composition patterns. Figure 2 depicts the automation of composite service according to the transactional rules presented in Table 2, Table 3, Table 4, and Table 5. The automaton consists of seven states. State I represents the initial state. Final states are p, c, cr, r, pr, and a. The alphabets of the language accepted by the automaton are {'p','pr','c','r','cr','a',' → p',' → pr',' → c',' → r',' → cr',' → a','+p','+pr','+-c','+r','+cr','+a','xp','xpr','xc','xr','xcr','xa'}.

For example, we consider the workflow shown in Figure 1. Suppose that service S1 (CRS) is compensatable retriable (cr), S2 (HBS) and S3 (FBS) are compensatable (c), S4 (PS) is pivot (p), and S5 (TDES) and S6 (TDMS) are retriable (r). In this workflow, first, we process parallel composition, then choice composition, and finally sequential composition using automaton. The transactional property of the composition is as follows:

$$cr \rightarrow (c + c) \rightarrow p \rightarrow (rxr) = cr \rightarrow c \rightarrow p \rightarrow r = c \rightarrow p \rightarrow r = a \rightarrow r = a.$$

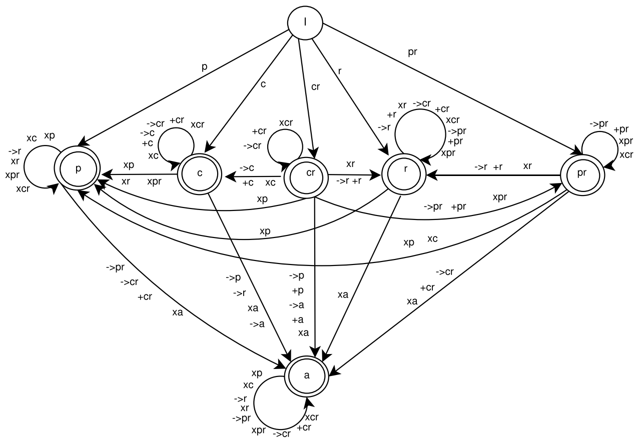


Figure 2. Automaton of Composite Service for transactional properties

Problem Statement

The main objective of web services composition is to build a reliable and efficient application, i.e., composite web service for a given workflow that satisfies transactional and QoS requirements.

The workflow WF represents specification of an abstract composite service. QR and TR are given QoS requirements and transactional requirements, respectively.

QR = {qr_1, qr_2, ...qr_m}, where qr_i represents the QoS requirement for the ith QoS attribute q_i and m is the number of QoS attributes.

TR = {tr_1, tr_1, . . . tr_n}, where tr_i is the transactional requirement of the ith abstract component service in the workflow and n is the total number of abstract component services or tasks in the workflow. The objective is to build a composite web service CWS with workflow WF satisfying QR and TR.

Related Work

Web services composition approaches can be categorized into three groups: 1) QoS-driven approaches, 2) transaction-driven approaches, and 3) transactional and QoS-driven approaches.

QoS-driven Approaches

This group of approaches focuses on the construction of composite web services based on the QoS requirements. Existing QoS-driven web services composition approaches are either scalarization-based or Pareto-based. In the scalarization-based approach, QoS attributes are normalized and weights are assigned to QoS attributes such that the sum of weights of QoS attributes must be equal to 1. Each composition solution is assigned a QoS score using the fitness function. This approach has some limitation. Weights are deduced by an expert based on user preference. Different experts may specify different sets of weights. The solution having the highest score is preferred, but there is no way to verify that the obtained solution is nondominated. In the Pareto-based approach, a set of solutions is obtained when solving the multiobjective optimization problem. These solutions define the best trade-off between conflicting objectives (QoS attributes), and this set of solutions is called the Pareto optimal set.

Researchers proposed exact, heuristic, and metaheuristic algorithms for web services composition. In Zeng et al. (2004), the authors used local and global optimization algorithms for QoS-aware web services composition. The local optimization algorithm chooses the optimal service for each task defined in the composite web service. The global optimization algorithm uses integer programming to select the optimal execution plan. Integer programming has polynomial time complexity and finds accurate results on the small size candidate service set. Chattopadhyay, Banerjee, and Banerjee (2017) proposed the dependency graph-based composition mechanism that finds service composition with near-optimal QoS in the least possible time and achieves scalability. Only a single QoS attribute can be considered at a time. The dependency graph-based approach has time complexity $O(n^2)$, where n is the number of services in the graph. Heuristic algorithms are generally experienced-based techniques created for solving specific optimization problems. Klein, Ishikawa, and Honiden (2011) proposed an approach based on a hill-climbing algorithm to obtain near optimal solution. Metaheuristic algorithms are problem independent and can be applied to a wide range of problems. In Jatoth, Gangadharan, and Buyya (2015), the authors presented the systematic review of existing research works on QoS-driven web services composition. They provided a classification of computational intelligence approaches to find web services composition based on QoS criteria and highlighted the future research challenges. The performances of well-known nature-inspired multi-objective algorithms are investigated in Cremene et al. (2016) to get optimal results for QoS-driven web services composition.

Canfora et al. (2005) proposed a Genetic Algorithm (GA)-based approach to obtain an optimal composite web service that satisfies global QoS requirements. Static and dynamic penalty strategies were used with the fitness function. This approach was compared with the integer programming-based approach presented in Zeng et al. (2004). The genetic Algorithm performs well with large search space in comparison to integer programming. Time compexity of GA is $O(mnt)$, where m is the number of component services in the workflow, n is the population size, and t is the number of generations. The nondominated Sorting Genetic Algorithm (NSGA-2) (Deb et al. 2002), a multiobjective evolutionary optimization algorithm, is used for service composition in Yao and Chen (2009). The time complexity of the proposed method is $O(mn^2t)$, where m is the number of QoS attributes, n is the population size, and t is the number of generations. In Liu et al. (2013), the authors applied end-to-end decomposition of QoS constraints to select web services for each component service of a workflow. The authors used the Culture Genetic Algorithm (CGA) that has better searching ability and fast convergence rate as compared to the Culture Max-Min Ant System. A hill climbing and genetic algorithm-based approach was proposed in Ai and Tang (2008) for QoS-aware web services composition that focused on both QoS constraints and dependencies between services. The optimzation approach accelerates the search for feasible solutions. Da Silva, Ma, and Zhang (2016) investigated three Genetic Programming (GP)-based service composition approaches. Each solution is represented as a tree. In the first approach, penalization in the fitness function is used to penalize functionally incorrect solutions. The second approach ensures functional correctness of the solutions by generating the initial population through a greedy algorithm. The third approach uses the second approach with the inclusion of the choice composition structure. The GP approach produces accurate composition solutions for large size repository of services in less time as compared to graph-based PSO. In Wu et al. (2016), the authors considered the concept of a Generalized Component Service (GCS) that could perform multiple abstract services. The standard parse tree-based algorithm was proposed to identify CGSs in a workflow of composite service. The authors used extended GA for finding a near-optimal composite web service solution. The GA-based approach finds a feasible solution with significantly less time for the number of component services > 10 than the backtracking algorithm. Karimi, Isazadeh, and Rahmani (2017) proposed clustering of services to decrease the search space of the problem so that discovery time can be reduced. SLA contracts were also clustered based on the similarity of the content of SLA contracts. The association rules technique is used in discovering the related clusters of different abstract services. The culture Genetic algorithm is used to find optimal solution for QoS-aware web services composition. Genetic algorithm-based optimization methods have the same time complexity as defined in Canfora et al. (2005).

In Zhang et al. (2010), the Ant Colony Optimization algorithm-based approach was proposed to obtain optimal QoS-aware service composition on the fly. Wang, Huang, and Xie (2014) described an Adaptive Ant Colony Optimization (AAOC)-based approach to find an optimal solution for QoS-aware web services composition. They considered the trust degree as an attribute to update pheromone evaporation dynamically. AACO has more than 90% accuracy in getting better Pareto solutions. The time complexity of ant colony optimization-based approaches is $O(mnt)$, where m is the number of component services in the workflow, n is the number of ants, and t is the number of generations. In Da Silva et al. (2017), graph database is used to store dependencies among available services in repository. A subgraph was built from graph database when receiving a composition request. Each composition solution was represented as a GP tree. The optimization approach produces near optimal solutions in a reasonable time. Zhao et al. (2012) proposed an approach that uses discrete immune optimization with particle swarm optimization (IDIPSO). The approach uses clonal selection theory and PSO to find optimal solutions for web service composition. Clonal selection theory describes the proliferation of antibodies that recognize a specific type of antigen. In web service composition, antigen represents the optimization problem and antibody refers to a solution, i.e., position vector of a particle. The proliferation of antibody (solution) is based on the affinity value. IDIPSO finds much better composition solutions than PSO. In Yin et al. (2014), the authors proposed a hybrid multiobjective discrete particle swarm optimization algorithm (HMDPSO) for the SLA-aware service composition problem (SSC). The authors considered three user categories platinum, gold, and silver. The optimization method has time complexity $O(mn^2t)$, where m is the number of QoS attributes, n is the population size, and t is the number of generations. HMDPSO produces more accurate solutions than MOGA as it includes a local search strategy. In Yan et al. (2016), the authors proposed a Graph-Based Memetic algorithm that increases search ability by combining global search and local search. The memetic algorithm is an extension of GA that includes a search technique to reduce the premature convergence of GA. In Chandra et al. (2016), the authors presented a modified gray wolf optimizer to select a composite web service that satisfies global QoS constraints. Time complexty of the modified gery wolf optimizer is $O(mnt)$, where m is the number of component services in the work flow, n is the number of wolves, and t is the number of generations. Gavvala et al. (2019) proposed an Eagle Strategy with the Whale Optimization method to obtain a global optimum solution for QoS-driven cloud service composition.The optimization method maintains the balance between exploration and exploitation to avoid premature convergence and slow convergence. The time complexity of the method is the same as that of GA. In Wang et al. (2018), the authors proposed a two-phase optimization process for QoS-aware web services composition. In the first phase, the credible services are obtained by integrating QoS computation with credibility evaluation and weight calculation using fuzzy AHP. In the second phase, a modified cuckoo method is applied to find global optimum solution for QoS-aware service composition. The time complexity of the method is $O(mn^2t)$, where m is the number of QoS attributes, n is the population size, and t is the number of generations. In Ren et al. (2021), the authors presented a QoS-aware composition framework based on deep learning and attention mechanisms. The approach finds high QoS composition solutions in less time.

Transaction-driven Approaches

Business-to-Business applications require web services composition to achieve their functionalities. Web services composition-based applications involve long-running web transactions that combine several transactions performed by different web services. These applications' executions can fail because of unavailability or failure of any web service, machine failure, process cancellation, etc. Thus, long-running web transactions violate some of the traditional transactional ACID (Atomicity, Consistency, Isolation, and Durability) properties, generally atomicity and isolation. Transaction-aware web services composition approaches focus on building reliable composite web services.

Bhiri, Perrin, and Godart (2005) proposed a transactional approach for reliable web service composition by ensuring failure atomicity of a composite service needed by designers. The authors distinguished between control flow and transactional flow of a transactional composite web service (TCS). The notion of the Accepted Termination State (ATS) is used to represent the designer's requirements for failure atomicity. ATS contains termination states of a composite service in which the designer accepts its termination. The execution of composite service is correct if the termination state of composite service belongs to ATS. The authors defined a set of transactional validity rules to generate transaction properties that ensure the validity of composite web services regarding the specified ATS. The time complexity of the method is $O(mn)$, where n is the number of ats in the set ATS and m is the size of each ats. In Bhiri, Perrin, and Godart (2006), the authors proposed reliable web service composition based on the concept of transactional patterns. A transactional pattern is a workflow pattern augmented with transactional dependencies between component services. There are two types of dependencies defined in this paper: activation dependencies and transactional dependencies (compensation, cancellation, and alternative).

In Li, Liu, and Wang (2007), the authors focused on transactional support to compose and schedule web services having different transactional properties. The authors proposed rules to derive transactional properties of composite web service described using sequential, parallel, alternative, and loop workflow patterns. Liu et al. (2009) proposed a framework for specification, verification, and execution of fault tolerant composite web service. Framework defined fault handling logic using Event Condition Action (ECA) rules at design time according to an application's need. The authors considered four types of transactional web services: atomic, weak atomic, semantic-atomic, and pivot. A weak atomic service is not compensatable, and semantic atomic service is not cancelable. The authors summarized exception handling strategies for handling faults. The authors also proposed a simple transfer-based termination protocol for consistent termination of composite service when the service's fault is unrepairable.

Transactional and QoS-driven Approaches

Execution of composite service should fulfill transactional and QoS requirements irrespective of any failure or dynamic changes. In El Hadad, Manouvrier, and Rukoz (2010), the authors proposed a design-time selection algorithm that satisfies the user's preference expressed in terms of QoS criteria and transactional requirements. Transactional requirements are defined in terms of risk levels. QoS requirements are described by assigning weights to each quality criterion. A simple weighted sum technique is applied to give a quality score to each web service. The web service selection for a task depends on the transactional property of the web service selected for the previous task in the workflow. The selection algorithm has time complexity $O(mn)$, where m is the number of component services and n is the number of candidate web services for each componet service. Wu and Zhu (2013) converted a workflow model of composite service into a directed acyclic graph using web service discovery. The solution is the path from the starting node to the sink node in DAG. An ant colony optimization (ACO) was employed to find the best path in DAG satisfying QoS constraints and transactional requirements. Each web service is assigned a QoS score and transaction score. The QoS score is calculated using the sum of weights of QoS attribute values. The time complexity of the optimization method is the same as in Wang, Huang, and Xie (2014). In Ding et al. (2015), the authors proposed an optimal selection approach based on the transactional performance evaluation method and Genetic Algorithm with a penalty function. The transactional performance method calculates the execution time of transactional composite web services according to the workflow pattern and transactional properties of component web services. An individual (solution) in GA was represented by integer array encoding. Execution time and price QoS attributes were considered.

Abbassi et al. (2015) proposed a Genetic Algorithm-based approach for ATS- and SLA-aware web services composition. The authors considered the response time, availability, throughput, and cost QoS attributes. Acceptable Termination States (ATSs) define the transactional behavior of composite services that express which faults are acceptable, retriable, or recoverable. The authors used transactional fitness (TF) and SLA fitness (SF) in the fitness function. An array of integers represents a genome (solution). In Graiet et al. (2016), the authors introduced the concept of recovery capability (RC) to capture the recoverability level supported by web services. If the transactional property of a service is compensatable and retriable, then the RC value of the service is 1. This means that the service is perfectly recoverable. The authors proposed an adaptive service composition (ASC) approach to define or reconfigure composite services according to SLA and ATS constraints. The authors used the Genetic Algorithm to find a composition solution. The time complexity of genetic-based optimization methods defined in Abbassi et al. 2015; Ding et al. 2015; Graiet et al. (2016) is $O(mnt)$, where m is the number of component services in the workflow, n is the population size, and t is the number of generations. In Imed and Graiet (2017), the authors defined a hierarchical service composition model using a set of functional capabilities of abstract composite service. The authors used the concept of Required Efficiency Level (REL). They proposed an algorithm that decomposes global QoS constraints into local ones then verifies the reliability and efficiency of web services. The time complexity of the hierarchical composition model is $O(k * (n + m + p))$, where n represents the number of tasks in the workflow, m is the number of inner nodes in the tree, p is the number of web services, and k represents the number of QoS constraints.

Grey Wolf Optimization Methods

This section describes the use of gray wolf optimization and its variants in the different application areas.

Li et al. (2017) proposed a modified discrete gray wolf optimizer (MDGWO) algorithm to find the optimal threshold value for multilevel image segmentation. They describe improvement in the attack strategy of gray wolves. Locations of search agents are updated using the weighting method. MDGWO has better accuracy than GA, Differential Evolution (DE), andAnt bee colony optimization techniques. In Martin, Marot, and Bourennane (2018), the authors proposed an improved version of discrete gray wolf optimization that performs a random selection of a leader. The method is tested on various benchmark functions and shows better performance. Sharma et al. (2019) described a modified Grey Wolf Optimization (MGWO) for detection of Parkinson's disease in a patient. The modified method is used for selecting multiple features, and it applies machine learning techniques to the selected features.

The estimated accuracy of the optimization method is 94.83%. In Badawy et al. (2018), the authors presented a discrete version of gray wolf optimization (DGWO) for the reconstruction of the shredded document. The position of each search agent is updated using either a crossover operator or a mutation operator. The DGWO has a fast convergence rate as compared to the genetic algorithm. Lu et al. (2016) described multiobjective discrete GWO for handling scheduling problems of the welding process. They include a two-part representation for solution encoding and a modified search operator to ensure the solution's feasibility. The optimization method presents better results than NSGA-II and Strength Pareto Evolutionary Algorithm 2(SPEA2) in terms of convergence and quality of optimal solutions. Karasu and Saraç (2020) employed a multiobjective gray wolf optimizer with KNN to choose the most appropriate features for the classification of power quality disturbances. The overall estimated accuracy of the classification model is 99.26. In Altan, Karasu, and Zio (2021), the authors applied the GWO method to optimize the weighted coefficients of each intrinsic mode function (IMF) for obtaining the best wind speed forecasting model. The proposed wind speed forecasting model performs well for different data sets.

Existing transactional and QoS-driven approaches are scalarization-based, and they used a simple weighted sum technique for thw QoS score calculation. They explained the formula of QoS computation for composite web services and rules for determining the transactional properties of web services composition but did not provide any method for the same. The use of other nature inspired algorithms in this area is still missing. Thus, based on the above study, we propose the Multiobjective Discrete Grey Wolf Optimization-based approach for transactional and QoS-driven web services composition. The proposed approach presents a tree model of composition workflow and discusses QoS computation- and automaton-based transactional properties determination for web services composition.

Proposed Approach for Web Services Composition

The overview of transactional and QoS-driven web services composition is depicted in Figure 3. This approach focuses on a tree model of workflow to determine the QoS attribute value and transactional property of composite service, generating a candidate set for each component service and a multi-objective discrete gray wolf optimization approach for finding composite web service optimal in terms of transactional and QoS requirements.

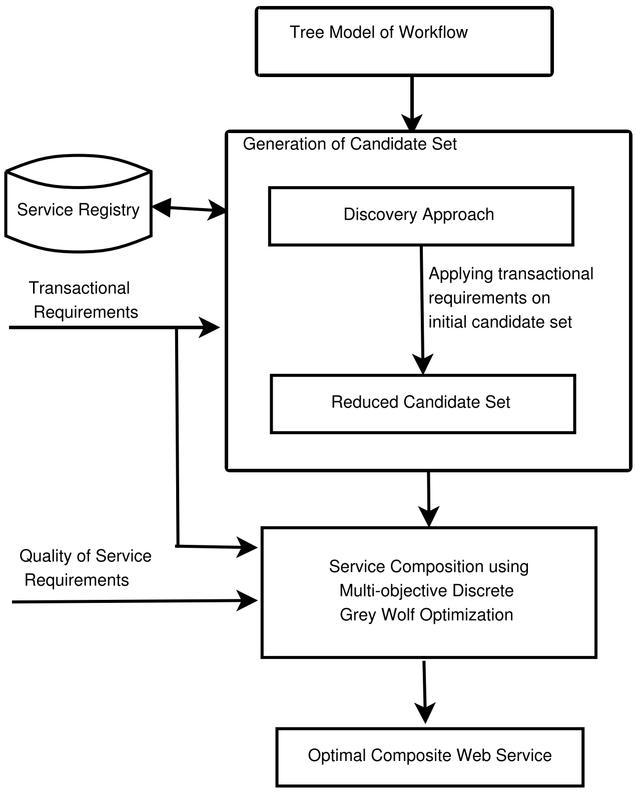

Tree Model of Workflow
Generation of Candidate Set
Service Registry
Discovery Approach
Applying transactional requirements on initial candidate set
Reduced Candidate Set
Transactional Requirements
Quality of Service Requirements
Service Composition using Multi-objective Discrete Grey Wolf Optimization
Optimal Composite Web Service

Figure 3. Proposed approach.

QoS requirements are specified in SLA based on mutual agreement between a service provider and a service requester. QoS requirements are constrained on the QoS attributes. For example, the response time of composite web service should not exceed 10 ms. We have considered the following constraints in the proposed approach.

- Response time of composite service $T_{cs} < = T_{max}$
- Cost of composite service $C_{cs} < = C_{max}$
- Reliability of composite service $R_{cs} > = R_{min}$
- Availability of composite service $A_{cs} > = A_{min}$

The application designer specifies transactional requirements at design time. For example, consider following the transactional requirements of OTMS.

(1) If either component service S2 or component service S3 fails during execution, then the effects of component service S1 must be semantically undone.

(2) If component service S2 fails and component service S3 completes, then the effects of component service S1 and component service S3 must be semantically undone.

(3) If component service S3 fails and component service S2 completes, then the effects of component service S1 and component service S3 must be semantically undone.

(4) If component service S4 fails during execution, it can be retried for its successful completion.

(5) The effects of component service S4 cannot be semantically undone after its successful execution.

(6) After completion of component service S4, either component service S5 or component service S6 will execute successfully.

Based on transactional requirements, the application designer decides transactional properties of component services so that the composition will execute correctly and consistently. Table 6 shows possible transactional properties of component services in OTMS to ensure failure-atomic execution.

Subsection 4.1 describes the tree model of workflow. The candidate set for each component service defined in the workflow is discovered using the web service discovery approach. It is described in subsection 4.2. A nature-inspired metaheuristic approach is used to solve multiobjective optimization problems.

Table 6. Transactional properties of component services in OTMS.

S1	S2	S3	S4	S5	S6
c	c	c	pr	pr	pr
c	c	c	pr	r	r
c	cr	cr	pr	pr	pr

In subsection 4.3, Grey Wolf Optimization is described. Subsection 4.4 explained multiobjective discrete gray wolf optimization to obtain web services composition optimal in transactional and QoS requirements.

Tree Model of Workflow

The workflow of a composite web service can be viewed as a generalized tree. We define a tree representation of workflow as WF_Tree = (root, leaf, qos, tp, type)

• root: It defines the root node of the tree. It contains the QoS values and transactional property of composite service.

• leaf: It is a boolean type. It tells whether a node is a leaf node or an internal node. Its value is true for a leaf node.

• qos: It stores the QoS value of an attribute of a node.

• tp: It stores the transactional property of a node.

• type: It gives the workflow pattern type such as sequence($\rightarrow$), parallel(+), choice(x), and loop(*) of an internal node other than root.

Figure 4 presents the tree model of workflow given in Figure 1. The root node represents the composite service, leaf nodes represent component services, and internal nodes represent different workflow patterns.

Algorithm 1 presents the computation of the QoS attribute value and transactional property for a composition workflow represented as a tree. It starts from the root node for each QoS attribute. It processes each child of the root. It stores the QoS value of each child in a list *qos_composite* and transactional property in a list *tp_composite*. If the QoS attribute is response-time or cost, then the QoS value of workflow is the sum of values defined in the list *qos_composite*. If the QoS attribute is reliability or availability, then the QoS value of workflow is the product of values defined in the list *qos_composite*. The transactional property of workflow is determined using automaton given in Figure 2 for *wf_pattern* and transactional properties defined in list *tp_composite*. Preprocessing of child node with parallel($+$) type, sequence ($\rightarrow$) type, loop($*$) type, and choice(x) type is given in Algorithm 2,

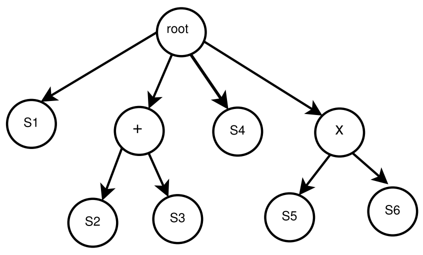


Figure 4. Tree model of composition workflow.

Algorithm 3, Algorithm 4, and Algorithm 5, respectively. The computational complexity of Algorithm 1 is O(M), where M is the number of nodes in the tree.

Generation of the Candidate Set

The workflow model specifies the compositional structure of a business application at the design time. Different tasks in the workflow represent different functionalities defined in the application. These tasks are connected through workflow patterns that define the execution order of tasks. Service providers publish descriptions of their services in a service registry. Thus, service registry may contain several services with similar functionality. The service requester can retrieve descriptions of services from the service registry using a discovery approach. Given a workflow WF of the composition, the orchestration engine uses web service discovery approach to discover a set of candidate web services for each task in WF. The web service discovery approach looks up in a service registry to find web services descriptions that match the functional requirements of a task in WF.

In our earlier work (Jalal, Yadav, and Negi 2019), we proposed a web service discovery approach incorporating Latent Dirichlet Allocation (LDA) and k-medoids clustering technique. The LDA topic model is used to extract topics from text corpus (Blei, Ng, and Jordan 2003). The k-medoids partition n number of objects into k number of clusters. The optimal value of k is obtained by silhouette analysis. The clustering technique increases the performance of web service discovery by reducing the search space for a user query. Web service discovery finds web services based on semantic similarity between web service description and user query. It uses WordNet database (Miller 1998) and Wu-Palmer method (Wu and Palmer 1994) to compute semantic similarity scores. After applying web service discovery, each component service defined in the workflow model may be associated with a set of candidate web services. For each component service, the size of the candidate set is reduced by considering transactional requirements. The reduced set contains web services that satisfy the transactional requirements of the application.

The Grey Wolf Optimization (GWO)

The Grey Wolf Optimization (GWO) (Mirjalili, Mirjalili, and Lewis 2014) is a new metaheuristic approach inspired by nature that depicts the leadership hierarchy of gray wolves and their hunting strategy for the prey. Grey wolves live in a group(pack) of 5–12 wolves.

```
Algorithm 1: Workflow_QoS_Transaction;
  Input : WF_Tree, root, attribute;
```

```
Output: composite_qos_val, composite_tp_val
1 get child_list of root node;
2 i=1;
3 while $i <= length(child\_list)$ do
4     child = child_list[i];
5     if $leaf[child] = True$ then
6         qos_composite[i] = qos[child];
7         tp_composite[i] = tp[child];
8     end
9     else
10        if $type[child] = "*"$ then
11            val1, val2 = Loop(child_list of child, attribute, k);
              // k is the number of loop iteration.
              // Loop module is defined in Algorithm 4.
12            qos_composite[i]= val1;
13            tp_composite[i]=val2;
14        end
15        if $type[child] = "+"$ then
16            val1, val2 = Parallel(child_list of child, attribute);
              // Parallel module is defined in Algorithm 2.
17            qos_composite[i]= val1;
18            tp_composite[i]=val2;
19        end
20        if $type[child] = "X"$ then
21            val1, val2 = Choice(child_list of child, prob_list);
              // prob_list contains probability of execution of each service.
              // Choice module is defined in Algorithm 5.
22            qos_composite[i]= val1;
23            tp_composite[i]=val2;
24        end
25    end
26    i=i+1;
27 end
28 if $attribute = "response\ time"$ or $attribute = "cost"$ then
29     composite_qos_val = $\sum$ qos_composite;
30 end
31 if $attribute = "reliability"$ or $attribute = "availability"$ then
32     composite_qos_val = $\prod$ qos_composite;
33 end
34 composite_tp_val = Automaton(tp_composite,wf_pattern=" $\rightarrow$ ");
   // Use automaton in Figure 2 for wf_pattern on transactional properties in tp_composite
35 return composite_qos_val, composite_tp_val
```

```
Algorithm 2: Parallel;
  Input : child_list, attribute;
  Output : qos_val, tp_val
  1 Create qos_par[], tp_par[];
  2 i=1;
  3 while $i <= length(child\_list)$ do
  4 if $leaf[child] = True$ then
  5 qos_par[i] = qos[child];
  6 tp_par[i] = tp[child];
  7 end
  8 else
  9 if $type[child] = "*"$ then
  10 val1, val2 = Loop(child_list of child, attribute);
  // Loop module is defined in Algorithm 4.
  11 qos_par[i]= val1;
  12 tp_par[i]=val2;
  13 end
  14 if $type[child] = " \rightarrow "$ then
  15 val1, val2 = Sequence(child_list of child, attribute);
  // Sequence module is defined in Algorithm 3.
  16 qos_par[i]= val1;
  17 tp_par[i]=val2;
  18 end
  19 end
  20 i=i+1;
  21 end
  22 if $attribute = "response\ time"$ then
  23 qos_val = $\max$ (qos_par);
  24 end
  25 if $attribute = "cost"$ then
  26 qos_val = $\sum$ qos_par;
  27 end
  28 if $attribute = "reliability"$ or $attribute = "availability"$ then
  29 qos_val = $\prod$ qos_par;
  30 end
  31 tp_val = Automaton(tp_par, wf_pattern="+");
  // Use automaton in Figure 2 for wf_pattern on transactional properties in tp_par
  32 return qos_val, tp_val
```

```
Algorithm 3: Sequence;
  Input : child_list, attribute;
  Output : qos_val, tp_val
```

```
1 Create qos_seq[], tp_seq[];
2 i=1;
3 while i <= length(child_list) do
4     if leaf[child] = True then
5         qos_seq[i] = qos[child];
6         tp_seq[i] = tp[child];
7     end
8     else
9         if type[child] = "*" then
10            val1, val2 = Loop(child_list of child, attribute,k);
              // Loop module is defined in Algorithm 4.
11            qos_seq[i]= val1;
12            tp_seq[i]=val2;
13        end
14    end
15    i=i+1;
16 end
17 if attribute = "response time" or attribute = "cost" then
18     qos_val = $\sum$ qos_seq;
19 end
20 if attribute = "reliability" or attribute = "availability" then
21     qos_val = $\prod$ qos_seq;
22 end
23 tp_val = Automaton(tp_seq,wf_pattern=' $\rightarrow$ ');
   // Use automaton in Figure 2 for wf_pattern on transactional properties in tp_seq
24 return qos_val, tp_val
```

Algorithm 4: Loop;

```
Input : child_list, attribute, k;
Output : qos_val, tp_val
// child_list contains a single element.
1 qos1 = qos[child];
2 tp1 = tp[child];
3 if attribute = "response time" or attribute = "cost" then
4     qos_val = product of k and qos1;
5 end
6 if attribute = "reliability" or attribute = "availability" then
7     qos_val = power(qos1,k);
8 end
9 tp_val = Automaton(tp1, wf_pattern="*");
  // Use automaton in Figure 2 for wf_pattern on transactional property tp1
10 return qos_val, tp_val
```

```
Algorithm 5: Choice;
  Input : child_list, prob_list;
  Output : qos_val, tp_val
  1 create tp_choice[];
  2 i=1;
  3 sum=0;
  4 while i< = length(child_list) do
  5 child = child_list[i];
  6 sum = sum + prob_list[i] * qos[child];
  7 tp_choice[i] = tp[child];
  8 i=i+1;
  9 end
  10 qos_val=sum;
  11 tp_val = Automaton(tp_choice, wf_pattern="X");
  // Use automaton in Figure 2 for wf_pattern on transactional properties in tp_choice
  12 return qos_val, tp_val
```

The alpha(α) wolves are the leaders and make all the decisions. Beta(β) wolves are next in the hierarchy and are responsible for advising and maintaining discipline. The next in the hierarchy are Delta(δ) wolves responsible for performing different tasks. The Omega(ω) wolves are lowest in the hierarchy and follow the instructions passed by alpha, beta, and delta wolves. The main steps of gray wolf hunting are tracking, encircling, and attacking the prey. Hunting is guided by alpha, beta, and delta wolves. The mathematical model of gray wolf optimization considers alpha(α) as the best solution and beta(β) and delta(δ) as second and third best solutions in the solution space. The encircling behavior of a gray wolf at iteration t +1 is defined in eq (1),

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}.\vec{D}, \tag{1a}$$

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)|, \tag{1b}$$

where t denotes the current iteration. $\overrightarrow{X_p}$ and $\overrightarrow{X}$ represent the position vectors of prey and gray wolf, respectively. $\overrightarrow{D}$ represents the difference in prey's position and the gray wolf's position at iteration t. The coefficient vectors $\overrightarrow{A}$ and $\overrightarrow{C}$ are given as follows:

$$\overrightarrow{A} = 2.\overrightarrow{a}.\overrightarrow{r_1} - \overrightarrow{a}, \tag{2}$$

$$\overrightarrow{C} = 2.\overrightarrow{r_2}, \tag{3}$$

where components of vectors $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ have random values in the range [0,1]. Each component of vector $\overrightarrow{a}$ is linearly reduced from 2 to 0 throughout the course of iterations. In a search space, the top three best solutions (α), beta(β), and delta(δ) have better knowledge about the optimum solution (position of the prey). Other search agents (grey wolves) update their positions according to the information provided by alpha (α), beta (β), and delta (δ) solutions (gray wolves). Hunting is mathematically formulated using the following equations:

$$\overrightarrow{D_\alpha} = |\overrightarrow{C_1}.\overrightarrow{X_\alpha}(t) - \frac{\$ \rightarrow \$}{X}(t)|, \overrightarrow{D_\beta} = |\overrightarrow{C_2}.\overrightarrow{X_\beta}(t) - \frac{\$ \rightarrow \$}{X}(t)|, \overrightarrow{D_\delta} = |C_3.\overrightarrow{X_\delta}(t) - \frac{\$ \rightarrow \$}{X}(t)|, \tag{4a}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha}(t) - \overrightarrow{A_1}.\overrightarrow{D_\alpha}, \overrightarrow{X_2} = \overrightarrow{X_\beta}(t) - \overrightarrow{A_2}.\overrightarrow{D_\beta}, \overrightarrow{X_3} = \overrightarrow{X_\delta}(t) - \overrightarrow{A_3}.\overrightarrow{D_\delta}, \tag{4b}$$

The updated location of the search agent at iteration (t +1) is given in equation (5),

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_2}}{3}. \tag{5}$$

Value |A|>1 or value |A|<-1 indicates that search agents (gray wolves) diverge from each other during the search for the prey (optimum solution). The value |A| < 1 indicates that gray wolves converge to attack prey. The optimum solution is found when termination criteria of hunting are satisfied. Vector $\overrightarrow{C}$ contains random values in the range [0,2]. It supports exploration and avoids local optima by emphasizing or deemphasizing the effect of prey.

Multiobjective Discrete Grey Wolf Optimization for Web Services Composition

The original GWO is designed to deal with a single objective optimization problem where search space is continuous. Transactional and QoS-driven web services compositions are a multiobjective combinatorial optimization problem and search space is discrete. QoS attributes are considered as different objectives. A standard multiobjective optimization problem can be defined as

$$F(x) = (F_1(x), F_2(X), \ldots, F_m(x)) \text{subject to} : \begin{cases} C_i(x) < = 0, i = 1, 2, \ldots, k, \\ C_j(x) = 0, j = 1, 2, \ldots, l. \end{cases}$$

Function F(x) has m objectives, and it can be either maximization or minimization subject to the various constraints C_i and C_j. Multiobjective optimization problems are Pareto dominant because they can have conflicting objectives. If objectives are conflicting, then improvement in one objective may cause the decline of other objectives. For example, in the case of web services composition, cost and availability are conflicting objectives. Any two solutions sol_1 and sol_2, are compared using Pareto dominance. Solution sol_1 dominates solution sol_2 if the following conditions are met:

- sol_1 is better than sol_2 in all objectives.
- sol_1 is better than sol_2 in at least one objective.

The resultant set, i.e., Pareto optimal set, contains nondominated solutions that ensure trade-offs in conflicting objectives. The Multiobjective Discrete Grey Wolf Optimization for web services composition is presented in Algorithm 6.

```
Algorithm 6: Composition-MDGWO;
  Input : Candidate Sets for Component Services;
  Output : Optimal Composite Web Service Solutions
  1 Initialize population P randomly using integer encoding
  2 foreach solution in P do
  3   Determine QoS values and the transactional property using Algorithm 1;
  4 end
  5 Initialize each componet of $\overrightarrow{a}$=2, and each component of $\overrightarrow{C}$=1 and
    maximum number of Iterations $t_{max}$;
  6 Initialize $\overrightarrow{A}$ using equation (2);
  7 Initialize t=0;
  8 while $t < t_{max}$ do
  9   Rank search agents (composite solutions) in P using nondominated
      sorting;
  10  Find $\alpha$, $\beta$, and $\delta$ solutions using equation (6);
  11  Store them in external archive;
  12  foreach search agent X(t) do
        // get new child population $P_{Ch}$
  13    Update search agent X(t) using equation (8);
  14  end
  15  t=t+1;
  16  $P_{temp}$ = Combined population P and population $P_{Ch}$;
  17  foreach solution in $P_{temp}$ do
  18    Determine QoS values and the transactional property using Algorithm 1;
  19  end
  20  Get population P for next iteration by applying nondominated sorting
      on $P_{temp}$;
  21  update each component of $\overrightarrow{a} = 2(1 - \frac{t}{t_{max}})$;
  22  update $\overrightarrow{A}$ using equation (2);
  23 end
  24 Find top 3 nondominated solutions from external archive;
```

25 return α, β and δ solutions

Solution Encoding

In web services composition, a composite web service instance, i.e., solution is represented by [S1, S2,..,Si,..,Sn]. It is encoded as an array of integers, and the length of the array is equal to the number of abstract component services in the composition workflow. The i^{th} element in the array represents the i^{th} abstract component service Si in the workflow. The value of the i^{th} element in the array is the web service index in the candidate set of the i^{th} abstract component service. Figure 5 shows an example of solution encoding for the web services composition. In this example, composite service consists of six abstract component services and each abstract component service has a candidate set of five web services.

Initial Population and Ranking of Solutions

The initial population is generated randomly based on the integer encoding representation. Algorithm 1 is used to compute the value of each QoS attribute and transactional property for each solution in the population. The multi-objective optimization problem can have a set of nondominated solutions. Nondominated sorting (Deb et al. 2002) is used to sort the solutions in population, and each solution has a rank equal to its nondominance level. Solutions are sorted on the basis of QoS attributes. At each nondominance level (or front), solutions can be ordered based on the score *Sol_Score* computed by equation (6),

$$Sol\_Score(S) = CD(S) + t\_score(S) - p\_score(S), \qquad (6)$$

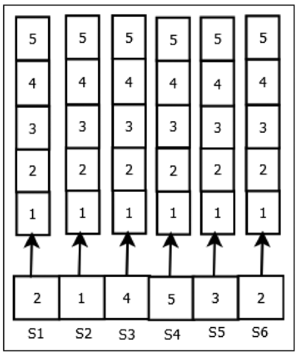


Figure 5. Example of solution encoding.

where S represents a solution in the population and $CD(S)$ is the crowding distance of the solution S. The value of t_score gives the transaction score of solution S, and it will be −1 if the transactional property of solution S is nonatomic. The value of $p\_score(S)$ is the penalty score of the solution S that is calculated using equation (7) as a difference from the given QoS requirements,

$$p\_score = \sum_{i=1}^{m} |qr_{\mathrm{i}} - q_{\mathrm{i}}| * x_{\mathrm{i}}, \tag{7}$$

$$where: \begin{cases} xi = 0 \ \ if \ q_{\mathrm{i}} satisfiesqr_{\mathrm{i}} \\ xi = 1 \ if q_{\mathrm{i}} \text{does not satisfy} qr_{\mathrm{i}} \end{cases}$$

A superior solution has a large value of crowding distance. The value of p_score will be zero for a solution if it satisfies all the QoS requirements.

Selection of Alpha, Beta, and Delta Solutions

The solutions at the first nondominance level are nondominated solutions for web services composition. If the population has only one nondominance level, then the best three solutions α, β, and δ can be obtained using Sol_Score values. If there are at least three nondominance levels, we choose a alpha solution from the first level, a beta solution from the second level, and a delta solution from the third level. After completion of each iteration, α, β, and δ solutions are stored in an external archive. When MDGWO terminates, the top three nondominated solutions can be selected from the external archive.

Searching and Hunting Prey

In discrete grey wolf optimization, we use two operator crossover for exploration and mutation for exploitation. Exploration is the process of searching for the prey, while exploitation is hunting the prey. A search agent performs crossover and mutation based on the value of $\overrightarrow{A}$. In discrete search space, the search agent (grey wolf or a solution) updates its position (information) using equation (8),

$$\overleftarrow{X}(t+1) = \begin{cases} crossover[\overleftarrow{X}_{\mathrm{p}}(t), \overleftarrow{X}(t)] & |\overleftarrow{A}| \geq 1 or |\overleftarrow{A}| - 1 \\ mutation[\overleftarrow{X}(t)]0 & \leq |\overleftarrow{A}| 1 \end{cases} \tag{8}$$

$\overrightarrow{C}$ is set to 1, and $\overrightarrow{A}$ is defined using equation (2). Information about the optimal solution (prey) is discovered through the knowledge of the α, β, and δ solutions. Thus, search agent X(t) updates itself by performing crossover with the best top three solutions α, β, and δ. Figure 6 depicts an instance of an update of search agent X(t). The newly generated population of offspring and parent population are combined. Then nondominated sorting technique and Sol_Score value are used to select the population of size N for the next

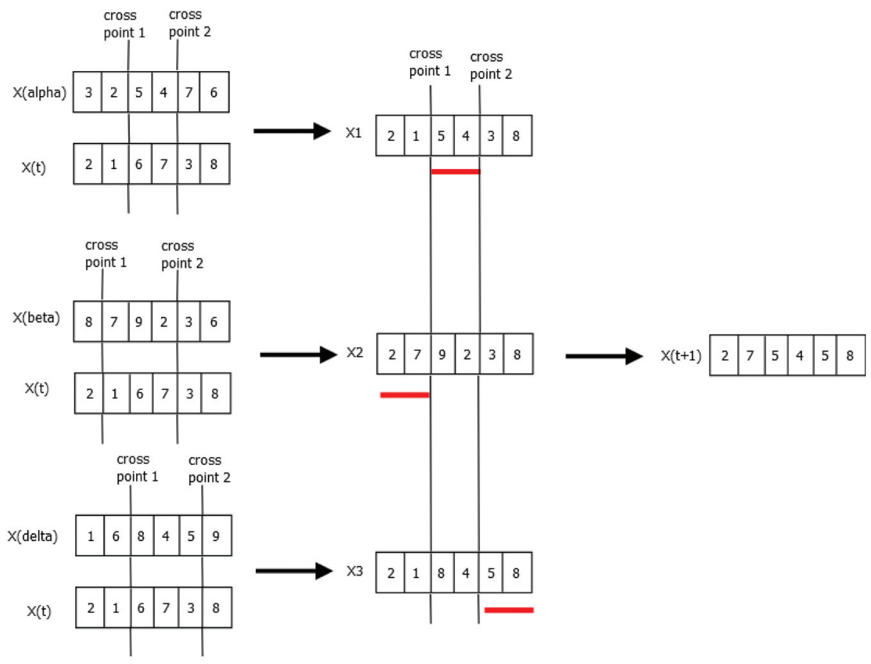


Figure 6. An instance of update of search agent X(t)

iteration. Time complexity of the proposed method is $O(mN^2t)$, where m is the number of QoS attributes, N is the population size, and t is the number of generations.

Experimental Setup

This section covers data set description, performance evaluation indicators, and result analysis. Performance of the web services composition based on Multiobjective Discrete Grey Wolf Optimization is compared with the well-known metaheuristic algorithms: NSGA-II (Deb et al. 2002) and Multiobjective Discrete Particle Swarm Optimization (MDPSO) (Yin et al. 2014).

Table 7. Data set information.

Domain	Customer Service	Hotel Service	Flight Service	Payment Service	Messaging Service
No. of Services	140	170	150	180	200

Data Set Description

We developed a data set of approximately 800 service descriptions from various domains such as Hotel, Flight Booking, Banking, and Finance published by the online web directory ProgrammableWeb (Musser 2005). Data are preprocessed using Natural Language Processing (NLP) functionalities provided by the NLTK toolkit (Loper and Bird 2002). Table 7 presents different domains and number of services in each domain.

We consider four QoS attributes: Response time, Cost, Reliability, Availability, and five transactional properties: pivot (p), compensatable (c), retriable (r), pivot-retriable (pr), and compensatable-retriable (cr). Web services in the data set are assigned values for QoS attributes and transactional properties using a random number generator with uniform distribution. TheiInitial candidate set (set of functionally relevant web services) for each task or domain is obtained using the web service discovery approach described in Jalal, Yadav, and Negi (2019). The reduced candidate set for each task is obtained by applying constraints defined by transactional requirements.

Performance Evaluation Indicators

Good convergence and maintaining diversity are desirable goals by evolutionary multiobjective optimization algorithms. The following indicators are used to evaluate the performance of multiobjective evolutionary algorithms NSGA-II, MDPSO, and MDGWO (proposed approach):

Generational Distance(GD)

The Generational Distance (Coello and Sierra 1999) indicator computes the distance between obtained Pareto front PF and the true Pareto front PF*. It is given as

$$GD = \frac{\sqrt{\sum_{j=1}^{N_1} (d_j)^2}}{N_1}, \tag{9}$$

where N_1 indicates the number of solutions presents on Pareto front PF and d_j is the euclidean distance between the j^{th} solution of PF and the closest solution of PF*. The lower value of GD indicates better convergence performance.

Inverse Generational Distance (IGD)

IGD (Sierra and Coello 2004) is a variant of the Generational Distance indicator. It computes the distances between each solution on the true Pareto front PF* and the closest solution on the obtained Pareto front PF. IDG is defined as

$$IGD = \frac{\sqrt{\sum_{j=1}^{N_2} (d_{\mathrm{j}})^2}}{N^2}, \tag{10}$$

where N_2 is the number of solutions present on true Pareto front PF*. The lower value of IGD shows good performance. IGD is used widely to assess the quality of an optimization algorithm that solves the problem with more than three objectives. GD and IGD are computationally fast.

Spread(S)

The spread indicator (Deb 2001) indicates diversity in the problem search space. It estimates the extent of the spread of solutions in the obtained Pareto front. It is defined as

$$S = \frac{\sum_{k=1}^{m} dx_{\mathrm{k}} + \sum_{j=1}^{N_1} |d_{\mathrm{j}} - \bar{d}|}{\sum_{k=1}^{m} dx_{\mathrm{k}} + N_1.\bar{d}}, \tag{11}$$

where $\bar{d}$ is the mean value of all d_{j} and m denotes the number of objectives. dx represents the distance of extreme solutions of obtained PF to the nearest solution of PF*. The lower spread value is desired for better diverse distribution. A zero value of spread indicates that all the solutions of PF* are uniformly spaced.

For several real problems, the true pareto front PF* is unknown. All non-dominated solutions obtained by different algorithms for a given problem instance can be considered as reference points on PF*. It is possible that reference points on PF* are not evenly distributed, i.e., some parts of the front may have very less reference points and other parts may have many reference points. We use the k-medoid clustering technique to group reference points into different clusters. The medoids of clusters can be considered as new reference points on PF*. A medoid is a most centrally located object in a cluster.

Result Analysis

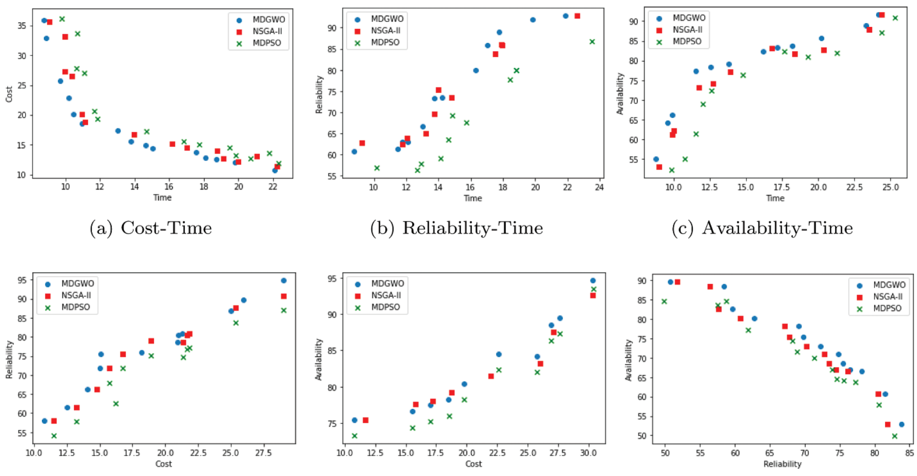
All the algorithms are implemented in the Python environment. Parameter settings can affect the performance of the optimization technique. The parameters are given in Table 8. All problem instances have the same values for

Table 8. Parameter information.

Problem	Component	Candidate	Pop-Size	Crossover	Mutation
Instance	Services	Set Size		Rate	Rate
Pb1	6	20	1000	0.85	0.15
Pb2	6	40	2000	0.85	0.15
Pb3	6	60	4000	0.85	0.15
Pb4	6	80	8000	0.85	0.15
Pb5	6	100	10000	0.85	0.15

parameters such as Component Services, Crossover Rate, and Mutation. Candidate Set Size and Pop_Size parameters are sensitive to the web services composition problem. Each problem instance has different values for Candidate Set Size and Pop_Size parameters. The population size of 1000 is appropriate for a Candidate Set Size of 20, but it is small for a Candidate Set Size of 100. Hence, the population size of the problem depends on the size of the candidate set for each component service. Each experiment was conducted with 20 independent runs on each problem instance for each optimization algorithm.

The results obtained for the proposed approach are compared with NSGA-2 and MDPSO and shown graphically in Figure 7 through (a) to (f). We have considered four QoS attributes and for each of QoS attribute pair, the comparison is carried out with the standard algorithms on problem instance *Pb*3. Figure 7(a) shows the set of nondominated solutions with respect to cost and response time QoS attributes. Similarly, Figure 7(b) compares each of the algorithms for a set of nondominated solutions with respect to reliability and response time. Figure 7(c-f) represent comparison with respect to availability-response time, reliability-cost, availability-cost, and availability-reliability pair QoS attributes, respectively. For each of the possible pairs of QoS attributes,



(a) Cost-Time (b) Reliability-Time (c) Availability-Time

(d) Reliability-Cost (e) Availability-Cost (f) Availability-Reliability

Figure 7. Nondominated solutions for two objectives.

Table 9. Pearson correlation coefficient values.

Algorithms	Cost Time	Reliability Time	Availability Time	Reliability Cost	Availability Cost	Availability Reliability
NSGA-2	−0.8704	0.9553	0.9220	0.9645	0.9516	−0.9710
MDPSO	−0.8734	0.9512	0.9164	0.9584	0.9481	−0.9603
MDGWO	−0.8581	0.9639	0.9100	0.9731	0.9545	−0.9632

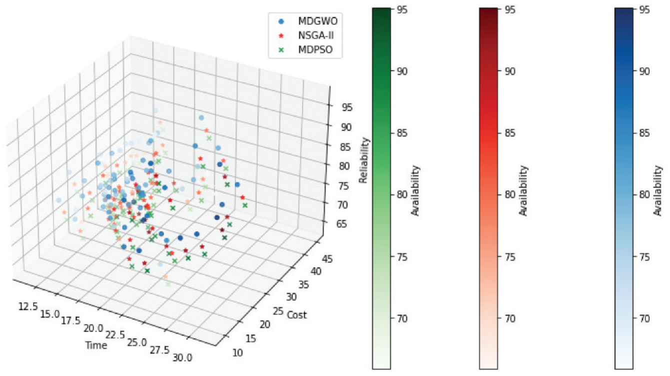


Figure 8. Nondominated solutions for four objectives.

the proposed approach reflects better results in terms of nondominated solutions. Correlation between different objectives is computed using the Pearson correlation coefficient. Table 9 presents the values of the Pearson correlation coefficient for various QoS attribute pairs for NSGA-2, MDPSO, and MDGWO algorithms.

The value of the Pearson correlation coefficient lies in a range from +1 to −1. A value of 0 indicates that the attributes are independent. Positive correlation implies that the value of one attribute rises so does the value of the other attribute. In contrast, a negative correlation suggests that the value of one attribute increases and the other attribute's value decreases. Figure 8 shows nondominated solutions obtained by MDGWO, NSGA-2, and MDPSO algorithms for four quality of service attributes on problem instance Pb3. A parallel coordinates plot on four QoS attributes in Figure 9 shows nondominated solutions obtained by the different algorithms on problem instance Pb3.

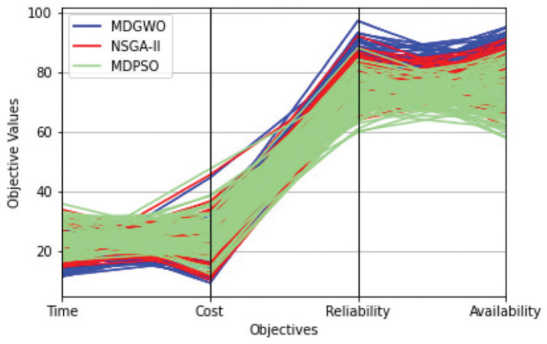


Figure 9. Parallel coordinates plot on four objectives.

Table 10. Indicators values obtained by NSGA-2, MDPSO, and MDGWO.

Problem Instance	NSGA-2 GD	MDPSO GD	MDGWO GD	NSGA-2 Spread	MDPSO Spread	MDGWO Spread	NSGA-2 IGD	MDPSO IGD	MDGWO IGD
Pb1	0.248	0.361	0.125	0.152	0.201	0.128	0.253	0.365	0.240
Pb2	0.365	0.499	0.279	0.191	0.243	0.132	0.386	0.452	0.377
Pb3	0.432	0.513	0.315	0.215	0.389	0.146	0.451	0.530	0.385
Pb4	0.590	0.699	0.478	0.351	0.429	0.237	0.593	0.694	0.481
Pb5	0.621	0.745	0.509	0.523	0.682	0.301	0.630	0.752	0.511
mean	0.451	0.563	0.346	0.286	0.388	0.189	0.462	0.558	0.398
std	0.139	0.140	0.132	0.135	0.169	0.0687	0.137	0.145	0.095

Table 11. Statistical results using the Wilcoxon signed rank test for MDGWO vs NSGA-2.

	GD			Spread			IGD		
	MDGWO vs NSGA-2			MDGWO vs NSGA-2			MDGWO vs NSGA-2		
Problem	R^+	R^-	*p* value	R^+	R^-	*p* value	R^+	R^-	*p* value
Instance									
Pb1	210	0	0.8e-04	189	21	0.16e-04	179	31	0.57e-02
Pb2	204	6	2.2e-04	200	10	3.8e-04	171	39	1.3e-02
Pb3	207	3	1.4e-04	204	6	2.2e-04	200	10	3.8e-04
Pb4	210	0	0.8e-04	210	0	0.8e-04	210	0	0.8e-04
Pb5	210	0	0.8e-04	210	0	0.8e-04	210	0	0.8e-04

Table 12. Statistical results using the Wilcoxon signed rank test for MDGWO vs MDPSO.

	GD			Spread			IGD		
	MDGWO vs MDPSO			MDGWO vs MDPSO			MDGWO vs MDPSO		
Problem	R^+	R^-	*p* value	R^+	R^-	*p* value	R^+	R^-	*p* value
Instance									
Pb1	210	0	0.8×10^{-04}	207	3	1.4×10^{-04}	207	3	1.4×10^{-04}
Pb2	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}	204	6	2.2×10^{-04}
Pb3	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}
Pb4	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}
Pb5	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}	210	0	0.8×10^{-04}

The comparison of MDGWO, NSGA-II, and MDPSO is shown in Table 10 using obtained statistical outcomes of indicators GD, IGD, and Spread for all the QoS attributes under consideration.

It can be observed from Table 10 that MDGWO performs well for transactional and QoS-driven web services composition. MDGWO maintains a social hierarchy among solutions in the search space. It has a good convergence rate because search for optimal solution is guided by best three solutions. The selection of the best three solutions from different Pareto fronts maintains population's diversity in generations and balances the exploration and exploitation processes.

Due to the stochastic nature of all optimization algorithms, a Wilcoxon signed rank test is used to discover the significant difference between the outcomes obtained by NSGA-II, MDPSO, and MDGWO algorithms on each problem instance with 20 independent runs. The significance level for all experiments is set to $\alpha = 0.05$. Tables 11 and 12 summarize the test results based on GD, Spread, and IGD indicators for MDGWO versus NSGA-II and MDGWO versus MDPSO, respectively. R^+ is the sum of ranks for the independent runs of each problem instance in which the MDGWO performed better than the compared one and R^- represents the sum of ranks for the compared one. It can be observed from Table 11 and Table 12 that MDGWO exhibits a high value of R^+ for GD, Spread, and IGD on each problem instance. This indicates that the MDGWO performs better than the two compared optimization algorithms.

Conclusion

In this work, we proposed an approach for Transaction-QoS driven web services composition. A tree model of workflow is described to determine the QoS attribute value and transactional property of composite service. We used web service discovery to retrieve functionally relevant web services (candidate set) for each task defined in the business application workflow model. Transactional requirements are used to reduce the size of the candidate set. A nature-inspired metaheuristic algorithm, multiobjective discrete gray wolf optimization, is used to select a composite web service solution that fulfills the transactional and QoS requirements. We compared the proposed approach with NSGA-II and MDPSO. Generational Distance(GD), Inverse Generational Distance(IGD), and Spread measures evaluate the web services composition approach. Evaluation results show that the proposed approach works well. The Wilcoxon signed rank test presents the significance of the proposed approach in comparison to NSGA-II and MDPSO.

The initial population plays a significant role in solving problems by evolutionary algorithms as it may impact the quality of the results obtained by evolutionary methods. The limitation of the proposed method is that the initial population is generated randomly. A portion of the initial population can be generated based on some heuristics to ensure the feasibility of the solutions. The proposed method also does not support web services composition on the fly. The applications of the proposed method can be possible in other domains, such as task scheduling for resources and selection of components from the library for component-based software engineering. According to the application, it may require making changes in the encoding and fitness computation of solutions.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

ORCID

Sunita Jalal http://orcid.org/0000-0001-7279-5971

References

Abbassi, I., M. Graiet, W. Gaaloul, and N. B. Hadj-Alouane (2015). Genetic-based approach for ats and sla-aware web services composition. In *International Conference on Web Information Systems Engineering*, pp. 369–83. Springer, Miami, FL, USA.

Ai, L., and M. Tang (2008). Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm. In *2008 IEEE Fourth International Conference on eScience*, pp. 119–26. IEEE, Indianapolis, IN, USA.

Alonso, G., F. Casati, H. Kuno, and V. Machiraju. 2004. Web services. In *Web services 2004: Data-centric systems and applications*, 123-149. Berlin: Springer.

Altan, A., S. Karasu, and E. Zio. 2021. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. *Applied Soft Computing* 100:106996. doi:10.1016/j.asoc.2020.106996.

Badawy, H., E. Emary, M. Yassien, and M. Fathi (2018). Discrete grey wolf optimization for shredded document reconstruction. In *International Conference on Advanced Intelligent Systems and Informatics*, pp. 284–93. Springer, Cairo, Egypt.

Bhiri, S., O. Perrin, and C. Godart (2005). Ensuring required failure atomicity of composite web services. In *Proceedings of the 14th international conference on World Wide Web*, pp. 138–47. ACM, Chiba, Japan.

Bhiri, S., O. Perrin, and C. Godart (2006). Extending workflow patterns with transactional dependencies to define reliable composite web services. In *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pp. 145–145. IEEE, Guadeloupe, French Caribbean.

Blei, D. M., A. Y. Ng, and M. I. Jordan. January 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Canfora, G., M. Di Penta, R. Esposito, and M. Villani (2005). An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 1069–75. ACM, Washington DC, USA.

Chandra, M., A. Agrawal, A. Kishor, and R. Niyogi (2016). Web service selection with global constraints using modified gray wolf optimizer. In *2016 International Conference on Advances in Computing*, Communications *and* Informatics (ICACCI), pp. 1989–94. IEEE, Jaipur, India.

Chattopadhyay, S., A. Banerjee, and N. Banerjee. 2017. A fast and scalable mechanism for web service composition. *ACM Transactions on the Web (TWEB)* 11 (4):1–36. doi:10.1145/3098884.

Coello, C. A. C., and M. R. Sierra. 1999. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations, In CINVESTAV-IPN (Evolutionary Computation Group); Departamento de Ingeniería Eléctrica, Sección de Computación, *Evolutionary computation* Citeseer, pp. 125-147, MÉXICO.

Cremene, M., M. Suciu, D. Pallez, and D. Dumitrescu. 2016. Comparative analysis of multi-objective evolutionary algorithms for qos-aware web service composition. *Applied Soft Computing* 39:124–39. doi:10.1016/j.asoc.2015.11.012.

Da Silva, A. S., H. Ma, and M. Zhang. 2016. Genetic programming for qos-aware web service composition and selection. *Soft Computing* 20 (10):3851–67. doi:10.1007/s00500-016-2096-z.

Da Silva, A. S., E. Moshi, H. Ma, and S. Hartmann (2017). A qos-aware web service composition approach based on genetic programming and graph databases. In *International Conference on Database and Expert Systems Applications*, pp. 37–44. Springer, Lyon, France.

Deb, K. 2001. *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley & Sons, Chichester, UK.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6 (2):182–97. doi:10.1109/4235.996017.

Ding, Z., J. Liu, Y. Sun, C. Jiang, and M. Zhou. 2015. A transaction and qos-aware service selection approach based on genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45 (7):1035–46. doi:10.1109/TSMC.2015.2396001.

El Hadad, J., M. Manouvrier, and M. Rukoz. 2010. Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. *IEEE Transactions on Services Computing* 3 (1):73–85. doi:10.1109/TSC.2010.5.

Gavvala, S. K., C. Jatoth, G. Gangadharan, and R. Buyya. 2019. Qos-aware cloud service composition using eagle strategy. *Future Generation Computer Systems* 90:273–90. doi:10.1016/j.future.2018.07.062.

Graiet, M., I. Abbassi, M. Kmimech, and W. Gaaloul. 2016. A genetic-based adaptive approach for reliable and efficient service composition. *IEEE Systems Journal* 12 (2):1644–54. doi:10.1109/JSYST.2016.2612641.

Imed, A., and M. Graiet. 2017. An automatic configuration algorithm for reliable and efficient composite services. *IEEE Transactions on Network and Service Management* 15 (1):416–29. doi:10.1109/TNSM.2017.2785360.

Jalal, S., D. K. Yadav, and C. S. Negi. 2019. Web service discovery with incorporation of web services clustering. *International Journal of Computers and Applications* 1–12. doi:10.1080/1206212X.2019.1698131.

Jatoth, C., G. Gangadharan, and R. Buyya. 2015. Computational intelligence based qos-aware web service composition: A systematic literature review. *IEEE Transactions on Services Computing* 10 (3):475–92. doi:10.1109/TSC.2015.2473840.

Jordan, D., J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, and A. Guızar. 2007. Web services business process execution language version 2.0. *OASIS Standard* 11 (120):5.

Karasu, S., and Z. Saraç. 2020. Classification of power quality disturbances by 2d-riesz transform, multi-objective grey wolf optimizer and machine learning methods. *Digital Signal Processing* 101:102711. doi:10.1016/j.dsp.2020.102711.

Karimi, M. B., A. Isazadeh, and A. M. Rahmani. 2017. Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm. *The Journal of Supercomputing* 73 (4):1387–415. doi:10.1007/s11227-016-1814-8.

Klein, A., F. Ishikawa, and S. Honiden (2011). Efficient heuristic approach with improved time complexity for qos-aware service composition. In *2011 IEEE International Conference on Web Services*, pp. 436–43. IEEE, Washington, DC, USA.

Li, L., C. Liu, and J. Wang (2007). Deriving transactional properties of compositeweb services. In *IEEE International Conference on Web Services (ICWS 2007*), pp. 631–38. IEEE, Salt Lake City, UT, USA.

Li, L., L. Sun, J. Guo, J. Qi, B. Xu, and S. Li. 2017. Modified discrete grey wolf optimizer algorithm for multilevel image thresholding. *Computational Intelligence and Neuroscience* 2017.

Liu, A., Q. Li, L. Huang, and M. Xiao. 2009. Facts: A framework for fault-tolerant composition of transactional web services. *IEEE Transactions on Services Computing* 3 (1):46–59. doi:10.1109/TSC.2009.28.

Liu, Z., X. Xue, J. Shen, and W. Li. 2013. Web service dynamic composition based on decomposition of global qos constraints. *The International Journal of Advanced Manufacturing Technology* 69 (9–12):2247–60. doi:10.1007/s00170-013-5204-6.

Loper, E., and S. Bird. 2002. *Nltk: The natural language toolkit*. arXiv preprint cs/0205028, Cornell University.

Lu, C., S. Xiao, X. Li, and L. Gao. 2016. An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. *Advances in Engineering Software* 99: 161-176. doi: 10.1016/j.advengsoft.2016.06.004

Martin, B., J. Marot, and S. Bourennane (2018). Improved discrete grey wolf optimizer. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 494–98. IEEE, Rome, Italy.

Mehrotra, S., R. Rastogi, H. F. Korth, and A. Silberschatz (1992). A transaction model for multidatabase systems. In *International Conference on Distributed Computing Systems*, pp. 56–63. IEEE, Yokohama, Japan.

Miller, G. A. 1998. *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.

Mirjalili, S., S. M. Mirjalili, and A. Lewis. 2014. Grey wolf optimizer. *Advances in Engineering Software* 69:46–61. doi:10.1016/j.advengsoft.2013.12.007.

Musser, J. (2005). Programmableweb.

Ren, X., W. Zhang, L. Bao, J. Song, S. Wang, R. Cao, and X. Wang (2021). Deepqsc: A gnn and attention mechanism-based framework for qos-aware service composition. In *2021 International Conference on Service Science (ICSS)*, pp. 76–83. IEEE, Xi'an, China.

Sharma, P., S. Sundaram, M. Sharma, A. Sharma, and D. Gupta. 2019. Diagnosis of Parkinson's disease using modified grey wolf optimization. *Cognitive Systems Research* 54:100–15. doi:10.1016/j.cogsys.2018.12.002.

Sierra, M. R., and C. A. C. Coello (2004). A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. *Technical Report of CINVESTAV-IPN.*

Statovci-Halimi, B., and A. Halimi. 2004. Qos management through service level agreements: A short overview. *e & i Elektrotechnik und Informationstechnik* 121 (6):243–46. doi:10.1007/BF03055357.

Van der Aalst, W., A. Ter Hofstede, B. Kiepuszewski, and A. Barros. 2003. Workflow Patterns. *Distributed and Parallel Databases* 14 (1):5–51.doi:10.1023/A:1022883727209.

Wada, H., J. Suzuki, Y. Yamano, and K. Oba. 2011. E^3: A multiobjective optimization framework for sla-aware service composition. *IEEE Transactions on Services Computing* 5 (3):358–72. doi:10.1109/TSC.2011.6.

Wang, D., H. Huang, and C. Xie (2014). A novel adaptive web service selection algorithm based on ant colony optimization for dynamic web service composition. In *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 391–99. Springer, Dalian, China.

Wang, H., D. Yang, Q. Yu, and Y. Tao. 2018. Integrating modified cuckoo algorithm and creditability evaluation for qos-aware service composition. *Knowledge-Based Systems* 140:64–81. doi:10.1016/j.knosys.2017.10.027.

Wu, Q., F. Ishikawa, Q. Zhu, and D. H. Shin. 2016. Qos-aware multigranularity service composition: Modeling and optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46 (11):1565–77. doi:10.1109/TSMC.2015.2503384.

Wu, Q., and Q. Zhu. 2013. Transactional and qos-aware dynamic service composition based on ant colony optimization. *Future Generation Computer Systems* 29 (5):1112–19. doi:10.1016/j.future.2012.12.010.

Wu, Z., and M. Palmer (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pp. 133–38, Las Cruces, New Mexico, USA.

Yan, L., Y. Mei, H. Ma, and M. Zhang (2016). Evolutionary web service composition: A graph-based memetic algorithm. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 201–08. IEEE, Vancouver, BC, Canada.

Yao, Y., and H. Chen (2009). Qos-aware service composition using nsga-ii1. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pp. 358–63. ACM, Seoul, Korea.

Yin, H., C. Zhang, B. Zhang, Y. Guo, and T. Liu. 2014. A hybrid multiobjective discrete particle swarm optimization algorithm for a sla-aware service composition problem. *Mathematical Problems in Engineering 2014:*14. doi:10.1155/2014/252934.

Zeng, L., B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. 2004. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering* 30 (5):311–27. doi:10.1109/TSE.2004.11.

Zhang, W., C. K. Chang, T. Feng, and H.-Y. Jiang (2010). Qos-based dynamic web service composition with ant colony optimization. In *2010 IEEE 34th Annual Computer Software and Applications Conference*, pp. 493–502. IEEE, Seoul, Korea (South).

Zhao, X., B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan. 2012. An improved discrete immune optimization algorithm based on pso for qos-driven web service composition. *Applied Soft Computing* 12 (8):2208–16. doi:10.1016/j.asoc.2012.03.040.