The International Journal of Advanced Manufacturing Technology (2019) 101:2771–2784
https://doi.org/10.1007/s00170-018-3028-0

ORIGINAL ARTICLE

A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal QoS-aware service composition and optimal selection in cloud manufacturing

Hamed Bouzary[1] · F. Frank Chen[1]

Received: 4 September 2018 / Accepted: 12 November 2018 / Published online: 11 December 2018

Abstract
Cloud manufacturing (CMfg), as a new service-oriented technology, is aiming towards delivering on-demand manufacturing services over the internet by facilitating collaboration among different producers with distributed manufacturing resources and capabilities. To this end, addressing service composition and optimal selection (SCOS) problem has been the pivotal challenge. This NP-hard combinatorial problem deals with selecting and combining the available resources into a composite service to meet the user's requirements while keeping up the optimal quality of service. This study proposes a new hybrid approach based on the recently developed grey wolf optimizer (GWO) algorithm and evolutionary operators of the genetic algorithm. The embedded crossover and mutation operators carry out a twofold functionality: (1) they make it possible to adapt the continuous structure of GWO to a combinatorial problem such as SCOS, and (2) they help to avoid the local optimal stagnation at the hunting process by providing more exploration strength. A series of experiments were designed and conducted to prove the effectiveness of the proposed algorithm, and the experimental results demonstrated that the proposed algorithm delivers superior performance compared with that of both existing discrete variations of GWO and genetic algorithm, especially in large-scale SCOS problems.

Keywords Cloud manufacturing · Service composition and optimal selection · Grey wolf optimizer · Metaheuristics · Quality of service · Industry 4.0

1 Introduction

The ever-changing business environment and customer expectations, such as high demand for personalized products, increased emphasis on immediate, responsive, and consistent service and a craving for keeping up with disruptive innovations, is among the present challenges the industry is confronted with. In order to address these challenges, producers need to be empowered to collaborate more effectively both inside their facilities and across different facilities [1]. These producers will then become partners working together through sharing their physical resources and technical capabilities to mutually increase their design, manufacturing, and marketing capacities and eventually achieve high customer satisfaction. Cloud manufacturing (CMfg) is such a paradigm through which dynamically scalable and virtualized manufacturing resources and capabilities are organized and controlled intelligently by cloud platform according to users' demands and offered to them on-demand as services over the Internet.

Achieving the ultimate goal of CMfg, i.e., constructing a fully integrated and collaborative environment for distributed manufacturing and eventually offering industrial resources and capabilities as services over internet (similar to what is now available in cloud computing for storage, processing power, or pay-as-you-go software) is not feasible without addressing the barriers associated with resource classification [2] and virtualization [3], task description [4, 5], associated enabling technologies [6–9], service discovery [10] and service matching [11], business models [12], trust evaluation [13], developing new architectures [14, 15], and service optimal selection

✉ F. Frank Chen
ff.chen@utsa.edu

[1] Department of Mechanical Engineering, The University of Texas at San Antonio, one UTSA circle, San Antonio, TX 78249, USA

and composition, among which the last one (service optimal selection and composition) seems to be most challenging. As briefly mentioned above, in a CMfg system, distributed manufacturing resources and capabilities are virtualized and encapsulated as services into manufacturing cloud services and then published in the resources pool through the Internet. Completion of a complex task submitted by a user needs the abovementioned collaboration among partners by selecting the optimal resource for each subtask out of the existing candidates at the resources pool and composing them to achieve the highest quality of service (QoS). Service composition and optimal selection (SCOS) deals with this particular problem in the context of cloud manufacturing.

Generally, manufacturing cloud services are unique service resources on the Internet. So, to evaluate such services on the Internet, the QoS criterion is typically used. The QoS represents a set of non-functional attributes of the service such as time, availability, throughput, cost, etc. QoS-aware SCOS deals with selecting the best service for each subtask from the corresponding candidate set in a way that results in the highest QoS value for the composed service in order to optimally deliver the complex task submitted by the customer. The QoS of composite service determines the efficiency of the collaboration among distributed partners as the number of these partners and their offered services grows, combining them into a composite service to meet the user's requirements while keeping up the optimal quality of service (QoS), becomes more crucial and exhaustive at the same time. Since the selected services in a composite manufacturing service have a mutual effect on the aggregated QoS value and this effect is nonlinear for some of the attributes (like availability and reliability), various metaheuristics have gained an increasing popularity for solving such large-scale NP-hard SCOS problems in the context of cloud manufacturing.

So far, a variety of metaheuristics from all three main classes (evolutionary, physics-based, and Swarm Intelligence) have been implemented to solve the SCOS problem. For instance, some variations of artificial bee colony (ABC) [16], genetic algorithm (GA) [17], particle swarm optimization (PSO) [18] and ant colony optimization (ACO) [19] are all leveraged for the problem resolution. Also, some hybridized algorithms have been successfully implemented and produced satisfactory results [20, 21]. However, according to the No Free Lunch theorem, it is logically implied that there is no particular algorithm that can obtain better results in all optimization problems. In other words, the performance of a particular metaheuristic is strongly dependent on the problem and nature of its search space. This justifies the never-ending enthusiasm of researchers regarding developing new metaheuristics and implementing them in real-world optimization problems.

Grey wolf optimizer (GWO) is a recently proposed swarm intelligence algorithm inspired by the social hierarchy found in the wolf packs and their haunting behavior [22] . Some of the recently reported applications include two-stage flow shop scheduling [23], optimal control of thermal systems [24], economic load dispatch problem of electric power systems [25], feature selection [26], and parameter selection in surface waves [27]. Generally, this algorithm has proved to produce competitive or superior results compared to other well-established algorithms such as PSO and DE (differential evolution), especially in terms of exploitation. This exploitative performance can be tracked down to its searching mechanism (called as "hunting") in which all wolves (solutions) update their positions according to the position of the best three wolves, thereby converging faster towards these good solutions. However, this exploitative mechanism makes the GWO algorithm prone to stagnation in local solutions. To overcome this premature convergence and achieve a better balance between exploration and exploitation, we propose incorporating mutation and crossover operators into the GWO, which eventually combines the merits of GWA and GA both. To the best of the authors' knowledge, this is the first implementation of GWO-based algorithms of any kind to solve the SCOS problem in cloud manufacturing systems.

The reminder of this paper is organized as follows. In Section 2, we review the most significant research works regarding service composition and optimal selection in cloud manufacturing context. In Section 3, the SCOS problem is described in a more detailed manner and then mathematically formulated in Section 4. Section 5 first briefly presents the standard grey wolf optimizer algorithm and then the proposed hybrid grey wolf optimizer algorithm with evolutionary operators is followed in a detailed manner. Section 5 verifies the effectiveness of our algorithm by addressing the designed experiments. Finally, Section 6 concludes the paper and proposes recommended future work.

2 Related work

As the demand for customization and personalized design grows, offering complex services at coarser granularities makes no sense anymore due to associated rigidity and lack of flexibility. Instead, focusing on providing finer single-functional services along with more robust composition mechanisms can bring more scalability and

elasticity through collaboration across various service providers. To assess and control the efficiency of this collaboration, QoS criteria is typically used and consequently, QoS-aware SCOS deals with selecting the best service for each subtask from the corresponding candidate set in a way that ensures obtaining highest overall QoS value for the composed service and subsequently, optimally delivering the complex task submitted by the customer. To this end, there is an ongoing stream of research work aiming for developing more efficient, robust and comprehensive mechanisms and algorithms to deal with the QoS-aware SCOS problem in cloud manufacturing systems. For the first time, Tao et al. [28] formulated this problem in the context of CMfg and then solved using an adaptive chaotic optimizer. With the advantages of making no or few assumptions about the solving problem, handling complex solution spaces, and obtaining near-optimal solutions within a reasonable time, various metaheuristics have attracted an increasing attention for solving the SCOS problem. In this regard, researchers have rarely utilized the basic form of these algorithms and have either innovatively modified their various phases or hybridized different algorithms to combine the merits of them. For instance, falling into the first category, Xiang et al. [29] integrated a new initialization mechanism based on case-library method into the GLA algorithm. Through case similarity compared with the user request, this initialization mechanism improved the accuracy and efficiency of the algorithm. As another example, Zhang et al. [30] reduced the search space through the concept of skyline service before using the core flower pollination algorithm, which successfully enhanced the quality of final solution and decreased the computational time. For the case of hybrid algorithms, Seghir and Khabala [21] proposed incorporating genetic operators into the fruit fly optimization algorithm to better the exploration ability. Zhou and Yao [20, 31] proposed a hybrid artificial bee colony algorithm with an improved perturbation mechanism in the employed bee phase and a Levy flight mechanism in the onlooker bee phase to solve the SCOS problem in cloud manufacturing systems and successfully reported an enhanced exploitation and convergence rate.

Aside from the solving algorithm, there is also an ongoing attempt to narrow down the gap between actual conditions associated with resource selection in CMfg platform and the modeling of the SCOS problem. For instance, Jin et al. [32] took into account the correlation among services through presenting a quality correlation-aware service description model for manufacturing cloud services and then proposed a service correlation mapping model to automatically obtain the associated QoS values. Li et al. [17] considered the correlations at the matching stage in order to prevent services with correlations from being left out during the matching process. Another stream of research is concerned with addressing dynamic service composition in CMfg in which change of service states imposed by the highly dynamic business environment is also taken into account. For instance, Liu et al. [33] approached this issue by dealing with the service composition and service scheduling as a unified problem. Yet another stream of work in this field is focused on releasing the assumption of a one-to-one mapping between services and resources, which will make it possible to perform each subtask collectively, thereby increasing the overall QoS and achieving more acceptable success rate. For instance, Liu et al. [34] proposed "Multi-Composition for Each Task" approach which successfully outperforms the existing one-to-one mapping composition approach. Yet another study by Liu and Zhang [35], based on the idea of combining multiple functionally equivalent elementary services into a synergic elementary service group, further improves the previous studies by achieving a higher fitness compared to Multi-Composition for Each Task and one-to-one mapping-based service composition methods. A detailed survey on service composition and optimal selection in cloud manufacturing systems by the authors of this paper can be found at Bouzary and Chen [36, 37].

3 Problem description

In every cloud manufacturing system, three main parties are involved in the completion of a manufacturing task, namely, service customer, service provider, and cloud platform. As customers request their services, providers publish and update their resources, including both physical manufacturing resources and capabilities, which have to be managed by the cloud platform. In other words, the service selection process happens through the cloud platform, which receives the requirements from cloud requesters and finds the most appropriate resources out of a pool of resources. Generally, the whole process of matching requests and resources can be categorized in three following steps, which are also shown in Fig. 1.

1. Task decomposition: First of all, the complex manufacturing task submitted by the customer has to be decomposed into several subtasks doable on a single-functional machine, equivalent to $T = \{ST_1, ST_2, ..., ST_i, ..., ST_I\}$, in which I is the total number of subtasks incorporating the task T and ST_i is the ith subtask of T. The QoS requirement of ST_i, which has to be determined by the service requester, is

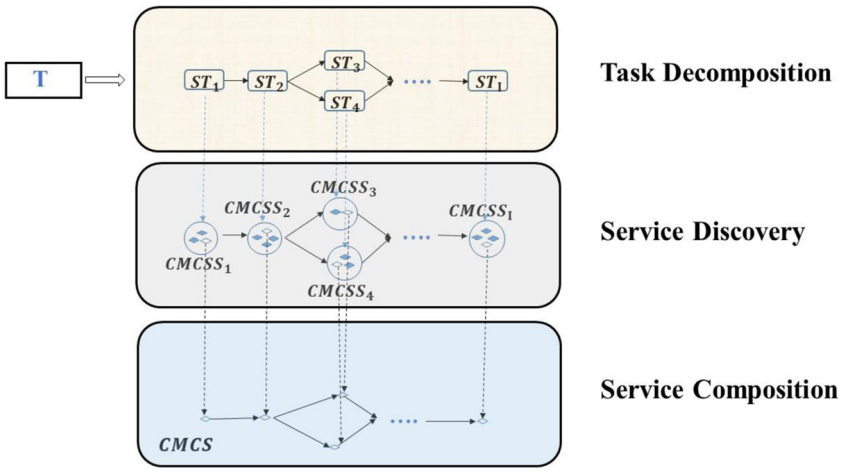

Fig. 1 Matching demands and resources in CMfg

denoted by $F(ST_i)$ while $F(ST_i) = \{f_{1i}, f_{2i}, ..., f_{ri}, ..., f_{Ri}\}$ in which q_{ri} is the rth requirement of ST_i and R is the number of considered QoS indexes [38].

2. Service discovery and matching: After decomposition, and based on functional requirements, a number of services will be retrieved for each subtask through putting into effect matching algorithms. These algorithms are mostly designed based on an evaluation of semantics similarity between the description of subtasks and resources on the cloud platform. In this way, each subtask can be associated with a set of candidate manufacturing cloud services (CMCSS). That is to say, $CMCSS_i = \{MCS_{i,1}, MCS_{i,2}, ..., MCS_{i,j}, MCS_{i,L_i}\}$ denotes the candidate set for ST_i where $MCS_{i,\ j}$ denotes the jth manufacturing cloud service in the set, and $L_i$is the total number of services in the ith candidate set ($CMCSS_i$) [38].
3. Service selection and composition: Finally, it is the time for identifying a service composition which can be defined as selecting one candidate service from each corresponding candidate set to compose the service while ensuring the overall QoS is optimal [9].

A more detailed explanation of the QoS service composition problem and its associated mathematical model will follow immediately.

4 Mathematical modeling

After the submission of task T to the cloud platform, and based on step 2, a number of MCSs are retrieved for each subtask ST and returned as a CMCSS. Now, we need to select one candidate MCS from each CMCSS to form a composite cloud manufacturing service (CCMS). A CCMS is mapped as one path. In fact, each complex task T can be mapped into different paths, which can be shown by $P = \{P_1, P_2, ..., P_m, ..., P_{L_{path}}\}$ where L_{path} is the number of possible paths, and $P_m$represents the m^{th} path of P. In this way, $P_m = \{MCS_{1,m1}, MCS_{2,m2}, ..., MCS_{i,mi}, ..., MCS_{n,mn}\}$ where each element denotes a possible composed solution for each of n decomposed subtasks of task T. QoS-aware service composition problem is defined as selecting the optimal composition path to execute task T while maximum QoS (built upon attributes like time, cost, reliability, availability, etc.) is achieved under customer's requirements which act as the constraints of this optimization problem.

Each candidate manufacturing cloud services set ($CMCSS_i$), contains m_i manufacturing candidate service (MCS). If N QoS indices are used to assess the candidates, then QoS of $CMCSS_i$ can be expressed as follows [38]:

$$F(CMCSS_i) = \begin{pmatrix} F(MCS_{i1}) \\ F(MCS_{i2}) \\ . \\ . \\ . \\ F(MCS_{im_i}) \end{pmatrix} = \begin{pmatrix} f_1(MCS_{i1}) & f_2(MCS_{i1}) & ... & f_N(MCS_{i1}) \\ f_1(MCS_{i2}) & f_2(MCS_{i2}) & ... & f_N(MCS_{i2}) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ f_1(MCS_{im_i}) & f_2(MCS_{im_i}) & ... & f_N(MCS_{im_i}) \end{pmatrix} \quad (1)$$

where m_i is the number of candidate MCSs in CMCSS and N is the number of QoS indices.

The QoS value of any composite CMfg service can be obtained through the aggregation of basic composition structures according to its execution path. The sequential, parallel, selective, and circular aggregation functions for four most used attributes are presented in Table 1.

The aggregated QoS values for each attribute should be normalized before calculating the global QoS of cloud manufacturing service, since each of these attributes has different measurement methods and dissimilar units and is either a positive or negative factor (for a positive factor, the bigger the value of the index, the better from the point of view of service requester and vice versa). This can be achieved using the Eqs. (2) and (3):

$$F_n = \begin{cases} \frac{f_n - \min f_n}{\max f_n - \min f_n} & if \quad \min f_n \neq \max f_n \\ 1 & if \quad \min f_n = \max f \end{cases} \tag{2}$$

$$F_n = \begin{cases} \frac{\max f_n - f_n}{\max f_n - \min f_n} & if \quad \min f_n \neq \max f_n \\ 1 & if \quad \min f_n = \max f \end{cases} \tag{3}$$

Equations (2) and (3) are associated with the normalization of positive QoS attributes (including availability and reliability), and negative QoS attributes (including time and cost), respectively.

Simple additive weighting (SAW) method is used to cover the four attributes simultaneously. Corresponding weights are multiplied by the attributes to obtain the QoS, as the final objective function. The optimal solution can be found by solving the following model:

$$Max \quad CF(P_m) = Max \sum_{1}^{N} w_k \times F_k(P_m) \tag{4}$$

s.t.

$$f_k\left(MCS_{i,mi}\right) \geq f_{k,i} \quad \text{if } f_k \text{is positive attribute} \tag{5}$$

$$f_k\left(MCS_{i,mi}\right) \leq f_{k,i} \quad \text{if } f_k \text{is negative attribute} \tag{6}$$

$$\sum_{k=1}^{N} w_k = 1 \tag{7}$$

where w_k is the weight of the kth index, N is the number of attributes, $f_{k,\,i}$ is customer's lowest required value of QoS for the kth index and $w_k \in [0, 1]$ [39]. Additionally, $f_k(MCS_{i,\,mi})$ is the QoS value of the m_ith MCS selected in the path P_m of the kth index. Moreover, $F_k(P_m)$ is the aggregated QoS values of MCSs associated with the kth index, which has been normalized according to Eqs. (2) and (3) and considering the structure of the path.

5 Algorithm design

As mentioned above, SCOS is an NP-hard and nonlinear problem and as a result, various metaheuristics have attracted the most attention. However, for three main reasons, the standard variations of these algorithms do not produce satisfactory results. First of all, the solution space of SCOS problem in the domain of cloud manufacturing is large and the QoS attributes affect the final composition interactively according to the complex task's execution path. Second, most of both well-established and recently developed algorithms are only suitable for continuous optimization problems, and hence they need to be modified before an adaption is made for a combinatorial problem such as SCOS. Last, but not least, some of the evolutionary and swarm-based algorithms such as GA and PSO suffer from easily trapping into a local mode which is mainly due to their ineffective evolutionary or learning mechanisms. With all these in mind, we design a new hybrid algorithm based on the recently developed "grey wolf optimizer" algorithm and by embedding evolutionary operators of crossover and mutation into its structure. In this way,

Table 1 QoS aggregation models

Structure	Cost	Time	Reliability	Availability
Sequence	$f_1 = \sum_{i=1}^{n} f_1(MCS_i)$	$f_2 = \sum_{i=1}^{n} f_2(MCS_i)$	$f_3 = \prod_{i=1}^{n} f_3(MCS_i)$	$f_4 = \prod_{i=1}^{n} f_4(MCS_i)$
Parallel	$f_1 = \sum_{i=1}^{n} f_1(MCS_i)$	$f_2 = Max\,(f_2(MCS_i))$	$f_3 = \prod_{i=1}^{n} f_3(MCS_i)$	$f_4 = \prod_{i=1}^{n} f_4(MCS_i)$
Selective	$f_1 = \sum_{i=1}^{n} (f_1(MCS_i)*\lambda_i)$	$f_2 = \sum_{i=1}^{n} (f_2(MCS_i)*\lambda_i)$	$f_3 = \sum_{i=1}^{n} (f_3(MCS_i)*\lambda_i)$	$f_4 = \sum_{i=1}^{n} (f_4(MCS_i)*\lambda_i)$
Circular	$\theta^* \sum_{i=1}^{n} (f_1(MCS_i)*\lambda_i)$	$\theta^* \sum_{i=1}^{n} (f_2(MCS_i)*\lambda_i)$	$f_3 = \prod_{i=1}^{n} f_3(MCS_i)$	$f_4 = \prod_{i=1}^{n} f_4(MCS_i)$

the algorithm would be fully adaptable for not only SCOS problem but also any other optimization problem with a discrete solution space. Moreover, our proposed algorithm not only makes it possible to implement GWO on combinatorial problems but also can provide a substitute for more simplified attempts regarding discretization of this algorithm [40].

5.1 Standard grey wolf optimizer

Grey wolf optimizer (GWO) is a recently-introduced swarm intelligence algorithm which was introduced by Mirjalai, et al. in 2014 [22]. Since then, it has gained a great level of attention and success and has been repeatedly implemented to solve various optimization problems. It mimics the leadership hierarchy and hunting mechanism of grey wolf packs in nature. Similar to other swarm-based metaheuristics, wolves navigate using the collective and social intelligence of the pack. The behavior of wolves is simulated through three main steps including hunting (searching for prey), encircling prey, and attacking prey. The pack is divided into four categories. The fittest solution is considered as the alpha (α). Second best and third best solutions are categorized as beta (β) and delta wolves (δ), respectively. The rest of the population is assumed to be omega (ω). Search for prey is led only by α, β, and δ wolves and ω wolves have to follow these three [22]. The mathematical interpretation of encircling prey, hunting, and attacking prey as the three pillars of GWO algorithm follows immediately.

5.1.1 Encircling prey

Grey wolves tend to encircle prey before the hunt. This behavior can be mathematically simulated through Eqs. (8–11).

$$\overrightarrow{D} = |\overrightarrow{C}.\overrightarrow{X}_p(t) - \overrightarrow{X}(t)| \tag{8}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A}.\overrightarrow{D} \tag{9}$$

$$\overrightarrow{A} = 2\overrightarrow{a}.\overrightarrow{r}_1 - \overrightarrow{a} \tag{10}$$

$$\overrightarrow{C} = 2.\overrightarrow{r}_2 \tag{11}$$

where t represents the current iteration, $\overrightarrow{X}_p$ indicates the position of the prey, and $\overrightarrow{X}$ represents the position vector of a wolf. $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors which make it possible for wolves to reach different locations around the prey. Essentially, D equals the distance of the wolf from the prey. Components of $\overrightarrow{a}$ linearly decrease from 2 to 0 over the course of iterations and r_1, r_2 are random vectors in [0,1] [22].

5.1.2 Hunting

Despite the fact that hunting is usually guided by the alpha, beta, and delta, in an abstract search space, the location of the optimum (prey) is unknown. In order to mathematically simulate the hunting process, it is assumed that α, β, and δ have better knowledge regarding the possible location of prey. Therefore, the position of three best individuals found so far is saved and position of every other wolf will be updated according to these three solutions. This can be achieved using Eqs. 12–14.

$$\overrightarrow{D}_\alpha = |\overrightarrow{C}_1.\overrightarrow{X}_\alpha - \overrightarrow{X}|, \overrightarrow{D}_\beta = |\overrightarrow{C}_1.\overrightarrow{X}_\beta - \overrightarrow{X}|, \overrightarrow{D}_\delta = |\overrightarrow{C}_1.\overrightarrow{X}_\delta - \overrightarrow{X}| \tag{12}$$

$$\overrightarrow{X}_1 = \overrightarrow{X}_\alpha - \overrightarrow{A}_1.\overrightarrow{D}_\alpha, \overrightarrow{X}_2 = \overrightarrow{X}_\beta - \overrightarrow{A}_2.\overrightarrow{D}_\beta, \overrightarrow{X}_3 = \overrightarrow{X}_\delta - \overrightarrow{A}_3.\overrightarrow{D}_\delta \tag{13}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X}_1 + \overrightarrow{X}_2 + \overrightarrow{X}_3}{3} \tag{14}$$

Through Eqs. (12–14), alpha, beta, and delta estimate the position of the prey and the other wolves update their positions randomly about the prey [22].

5.1.3 Attacking prey and searching for prey

The process of approaching the prey is mimicked through decreasing value of $\overrightarrow{a}$. This also fluctuates the $\overrightarrow{A}$ in the range of $[-2a, 2a]$ as $\overrightarrow{a}$ decreases from 2 to 0 over the course of iterations. In this way, while random values of $\overrightarrow{A}$ change between -1 and 1, the next position of a wolf can be somewhere between its current position and the position of the prey. On the other hand, any random value of $\overrightarrow{A} > 1$ or $\overrightarrow{A} < -1$ leads to an opposite behavior called "searching for prey" which provides better global search and helps to avoid getting stuck in locally optimal solutions.

Through the abovementioned operators, GWO guides its search agents to gradually update their positions based on the position of the best three agents and eventually obtain a near-optimal solution. A more detailed explanation on this algorithm can be found in [22].

5.2 Discrete hybrid grey wolf optimizer

5.2.1 Encoding and decoding

As mentioned before, the standard GWO is designed to solve optimization problems with continuous solution space, since the updated position of wolves could take any possible value within the solution space. However, the SCOS problem is a typical combinatorial problem. Therefore, a discrete hybrid

variation of grey wolf optimizer equipped with evolutionary operators has been developed to pave the way for making use of this powerful algorithm in the context of SCOS problem, and any other combinatorial problem in general. As our first step, we have to develop an encoding scheme according to the characteristics of the SCOS problem. To represent a solution, we use a single string with the number of digits equal to a number of subtasks. For instance, the representation in Fig. 2 encodes a composition in which the first subtask is assigned to candidate 3, second subtask is assigned to candidate 1, third subtask is assigned to candidate 7, fourth subtask is assigned to candidate 4, and finally, the fifth subtask is assigned to candidate 9. As one can see, each candidate has been chosen from a different set of candidates associated with each subtask.

5.2.2 Embedded evolutionary operators

Solving a combinatorial optimization problem is very different from solving a problem with a continuous solution space. In a continuous search space, the search agents of GWO are able to update their position vectors according to the hunting process mimicked by Eqs. (12–14). In a discrete space, however, the position of wolves cannot be updated likewise, since the position vectors can only take on discrete values. The first intuitive solution that comes to mind in these type of situations is using the floor function which takes the value of a position vector in each direction as an input and gives as output the greatest integer less than or equal to that. However, this method diminishes the algorithm's strength in both exploration and exploitation phases, as a result of lost variability caused by the abundance of repetitive individuals. Alternatively, and according to Mirjalili and Lewis [41], another technique to convert a continuous swarm-based algorithm to a binary algorithm without modifying the structure is to employ a transfer function, which is also applicable for discrete solution spaces. For instance, Emary et al. [26], took a similar approach to develop a discrete GWO algorithm through employing a sigmoid function to solve the feature selection problem. However, we did not find the proposed algorithm effective enough to solve the SCOS problem, mostly due to the fact that it did not provide enough explorative power for a large-scale problem like SCOS, and as a result, mostly got stuck in local optimal locations. This urged us to develop a modified algorithm based on the GWO by adding evolutionary operators into the structure of the standard algorithm. This approach can, to a great extent, improve the balance between exploration and exploitation, yet preserving the advantages of both GWO and GA algorithms. All the considered modifications intended to adapt GWO to the SCOS problem, as well as the added crossover and mutation operators are presented as follows.

Range checking Updating the position of wolves during the hunting process using Eqs. (12–14) does not guarantee that the value of every direction does not exceed the bounds of the problem. As a result, after attaining the discrete values of each direction through rounding towards zero, the updated values of positions obtained from Eqs. (12–14), an additional step is needed to be considered to return back the exceeding values into the boundaries of the problem. In the case of SCOS problem, upper bound denoted by *ub*, equals to the number of available candidates for each subtask, and lower bound, denoted by *lb*, is equal to 1. Equation (15) is applied to achieve this goal.

$$\vec{X}_i = \left\lfloor \vec{X}_i \right\rfloor . \left(\overline{u + l} \right) + ub.u + lb.l \tag{15}$$

where $u = \left\lfloor \vec{X}_i \right\rfloor > ub$ and $l = \left\lfloor \vec{X}_i \right\rfloor < lb$.

Selection operator Roulette wheel selection is a widely used selection operator applied in several proposed genetic algorithms solving the service composition problem [42]. The basic idea behind this operator is the fact that the better the fitness of an individual, the larger the probability of its survival and mating. The same selection operator is applied here to select the parents that will go through the crossover and mutation operators.

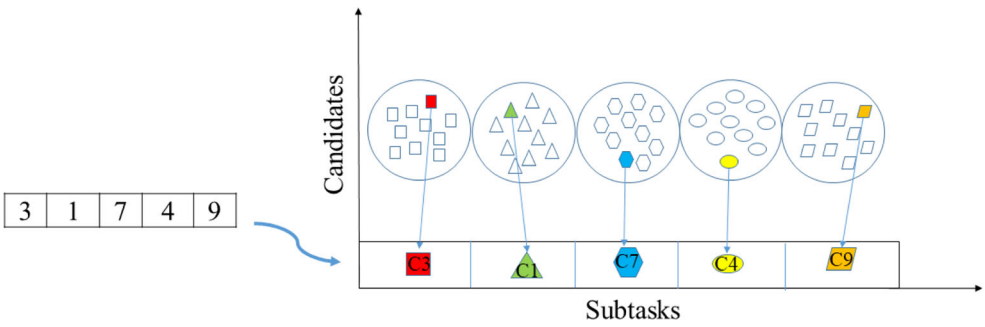

Fig. 2 An example of encoding and decoding for SCOS problem

To select an individual for crossover and mutation processes out of a population with size n (size of the wolf pack in our case) we use the following steps:

Step 1. Generate a random real number (r) ∈ [0, 1].
Step 2. If $r \le p_1$, select the first wolf in the population w_1, otherwise; select the ith wolf w_i ($i = 1,2,\ldots, n$) such that $p_{i-1} < r \le p_i$, where p_i is the cumulative probability for each wolf w_i in the current population (pack) which will be calculated according to Eq. (16).

$$\begin{cases} p_1 = \dfrac{f_i}{\sum_{j=1}^{n} f_j} \\ p_i = p_{i-1} + \dfrac{f_i}{\sum_{j=1}^{n} f_j} \quad if\ 2 \le i \le n \end{cases} \tag{16}$$

where f_i is the fitness of an individual i in the population [21].

Crossover operator Every selected pair of parents will go through the crossover operator and a pair of new individuals (children) will be produced by combining their genes. Here, we have employed a two-point crossover operator similar to some other research works on service composition [43]. During this operator, two crossover points are randomly chosen in the two parents. Then, the subsequent genes, between these two cutting points of each parent are exchanged from a chromosome to another one (see Fig. 3a).

Mutation operator Mutation operator is intended to provide exploration and to avoid local optimum stagnation through maintaining the diversity of population which will be achieved by making a small and slow genetic frequency change in the generated chromosomes. Swap mutation is used here in which two genes in the chromosome of an individual will be randomly selected and then their values will be interchanged (see Fig. 3b).

Now that crossover and mutation operators are applied, an overlapping population will emerge, where parents and offspring are merged, and the best individuals from this union will form the next population. In fact, this merged population will be sorted in descending order and the first best n individuals (equal to the size of the wolf pack) will be selected and replaced as the initial population for the next iteration. The whole process from the initialization up until this point will be repeated until a stopping criteria is met. Exceeding a specific CPU time, reaching a predefined number of iterations, a certain number of iterations without any improvement in the fitness of the fittest solution and converging to a specific fitness value are among the most popular termination criteria for metaheuristic algorithms. In this study, we have used the maximum number of iterations as the stopping criterion for our proposed algorithm. A summary of the embedded procedures of the proposed hybrid GWO algorithm is organized step by step in Algorithm 1.

6 Experiments and results

In this section, we investigate the performance and efficiency of the proposed hybrid GWO algorithm for SCOS problem and its potential for solving other combinatorial problems as well. To this end, this section conducts performance comparison with different metaheuristics under various testing scenarios as well as different user QoS preferences and QoS value ranges. In order to evaluate the performance of the algorithm for different SCOS problem sizes, the comparisons are conducted under two following scenarios:

(1) Scenario 1, in which the number of subtasks N remains fixed as 10, while the number of available candidate services M in each CMCSS various in the range of 50 to 500 with an increment of 50.
(2) Scenario 2, where the number of candidate services M is fixed as 50 for all the cases, while the number of subtasks varies from 5 to 20 with an increment of 5.

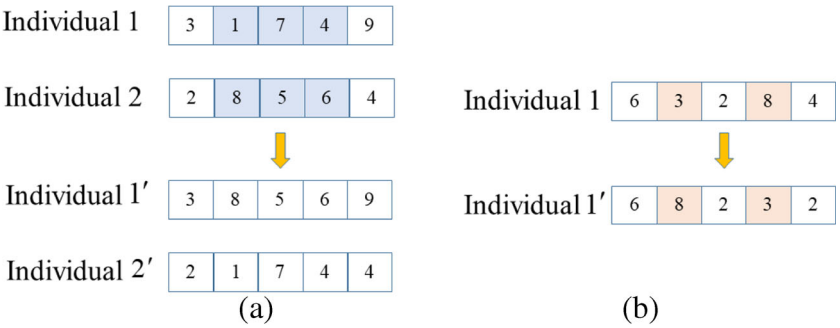

Fig. 3 Evolutionary operators in genetic phase. **a** Two-point crossover. **b** Swap mutation

Algorithm 1: Hybrid GWO algorithm for SCOS problem

(1) ***Initialize*** ($t = 0$)

Step 1: Randomly generate an initial population $X_i(i = 1, 2, ..., n)$, where n denotes the size of the wolf pack

(2) ***Repeat*** ($t = 1$ to *maxiter*)

Step 2: Initialize vectors $\vec{a}, \vec{A}, \vec{C}$ according to Eqs. (10–11)

Step 3: Calculate the fitness values of each wolf

Step 4: Record the position of first three best wolves as X_α, X_β and X_δ, respectively and update accordingly

Step 5: Update the position of wolves through hunting process using Eqs. (12–14)

Step 6: Discretize the elements of position vectors obtained in step5 $X \rightarrow t+1$ using the floor function

Step 7: Perform range checking for values obtained in Step6 and map out of bound values (if any) into the allowed space according to Eq. (15)

Step 8: Select $P_c \times n$ the number of parents for crossover operation and $P_m \times n$ the number of parents for mutation operator according to Eq. (16), where P_c and P_m are the probability of crossover and mutation, respectively

Step 9: Perform the two-point crossover on the selected parents

Step 10: Perform the swap mutation operator on the selected individuals for a mutation

Step 11: Merge the populations created in Steps 9 and 10 with the original population obtained after Step7

Step 12: Sort the step 11's population in descending order and only keep the first fittest n individuals.

Step 13: Replace this new population as the initial population for the next iteration

Step 14: Memorize the best solution achieved so far. Assign $t + 1 \rightarrow t$.

Step 15: If the step criterion is not reached go to step 2.

Step 16: Output the best solution

(3) ***Output***

Considering the fact that every composite MCS path, including sequential, parallel, circular, and selective connections, can eventually boil down into a certain combination of equational paths [44], only the sequential aggregation model is considered here without loss of generality. On the other hand, in order to prove the superior performance of the proposed hybrid GWO algorithm, it is compared with an existing discrete variation of GWO algorithm introduced by Emary et al. [26], in which the algorithm is implemented to solve the feature selection problem. In fact, Emary et al. developed a binary variation of the GWO algorithm using a logistic transfer function and here we have adapted the same idea to achieve a discrete algorithm using the following function:

$$\begin{cases} \text{Update the associated direction of the individual according to Eq.(14)} & \text{if } sigmoid\left(\dfrac{X_1 + X_2 + X_3}{3}\right) \geq rand \\ \text{Do not change the value of associated direction} & otherwise \end{cases} \tag{17}$$

Equation (17) is adapted from the Eq. (24) of [26] (other steps of the algorithm are implemented exactly according to algorithm 3 from page 374 of [26]). From now on, this algorithm is going to be denoted as DGWO in this paper. In addition, GA, as a well-established and widely used intelligent algorithm to solve the SCOS problem, is also implemented and compared with the proposed approach. Without loss of generality, all of the QoS values of available candidates are randomly generated with a uniform distribution within the range [0.7, 0.9]. Four QoS attributes of time, cost, reliability, and availability are considered to build the objective function, while their weights (w_1, w_2, w_3, and w_4, respectively) are all set to 0.25. Having the stochastic nature of metaheuristic algorithms in mind, each experiment has been run 10 times and the average and standard deviation are reported to provide a fair comparison. The proposed and compared algorithms are coded in MATLAB R2013b on a computer with 2.3 GHz processor and 8 GB of RAM and a 64-bit Windows 10 operating system. As one can see from Table 2, grey wolf optimizer has the advantage of needing very few parameters to be tuned. The special parameters of these three algorithms are set as Table 2.

Table 2 Parameter settings

Algorithm	Initial population	Crossover probability	Mutation probability	x-value of Sigmoid's midpoint (x_0)	Steepness of the logistic function (k)
HGWO	20	0.15	0.85	–	–
DGWO	20	–	–	0.5	10
GA	20	0.15	0.85	–	–

6.1 Performance of the proposed hybrid GWO algorithm (HGWO) compared with DGWO and GA

As our first evaluation, the aggregated fitness values of the best-composed services are obtained according to scenario 1 and the average and standard deviation values are reported as each experiment has been repeated 10 times. As one can see from Fig. 4a, the average values for HGWO slightly decreases as the number of available candidates M increases from 50 to 150 and also from 150 to 300, which can generally be attributed to the fact that as the size of the solution space grows, the probability of finding an optimal solution drops. However, the results indicate a positive trend over the span of 150–200 M (number of candidates), which can mark the optimal size of SCOS problem to be solved by our proposed HGWO algorithm as $M = 200$, when N (number of subtasks) is equal to 10. Moreover, in all of the scenario 1 cases, HGWO outperform GA and DGWO in terms of the obtained optimal QoS value and this superiority compared with DGWO, becomes more notable in the large-scale problems (average fitness value equal to 0.65 for HGWO, versus value of 0.41 for DGWO in case of $M = 300$ and $N = 10$). In addition, GA proves a more stable performance as the number of candidates increases and according to Fig. 4a, DGWO is the one that experiences the most significant deterioration in performance as we move towards large-scale problems.

According to Fig. 4b, the optimal QoS fitness values obtained by all three algorithms, decrease drastically as the number of subtasks increases. However, HGWO still outperforms the others with an average fitness of 0.61, followed by GA which has obtained an average fitness of 0.52 and DGWO with the worst performance among them with an average QoS equal to 0.48. Comparing a and b in Fig. 4, we can notice that the performance of these algorithms is heavily influenced by an increment in the number of subtasks, while that does not change obviously and meaningfully as the number of available candidates goes up. This can be due to the fact that the size of the solution space associated with SCOS problem can be obtained through an exponentiation operation b^n, where the base b is equal to the number of available candidates M, and the exponent n would be equal to the number of subtasks N. As a result, any increase in the number of subtasks will lead to a bigger impact on the size of the solution space and subsequently, a higher reduction in the probability of finding an optimal solution in a certain number of iterations.

In order to investigate the efficiency of the proposed algorithm, CPU consumption time associated with both scenario 1 and scenario 2 are recorded and the average amounts are presented in Fig. 5a, b, respectively. It can be seen that, generally, GA has the least computational time, followed by DGWO and

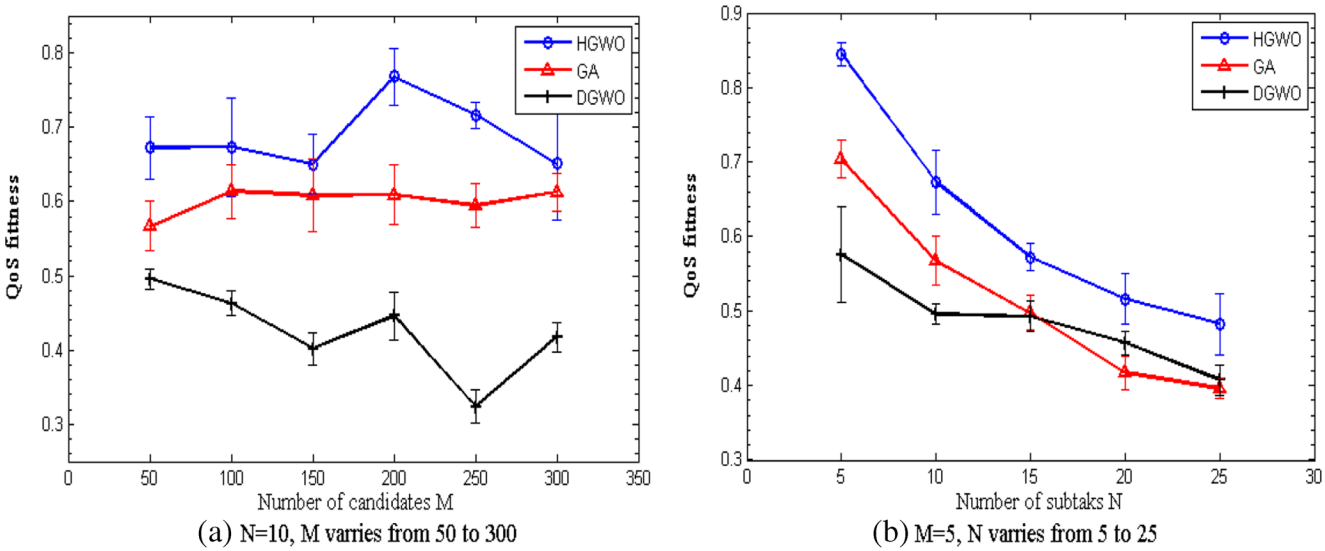

Fig. 4 The proposed HGWO compared with DGWO and GA for different scales. **a** $N = 10$, M varies from 50 to 300. **b** $M = 5$, N varies from 5 to 25

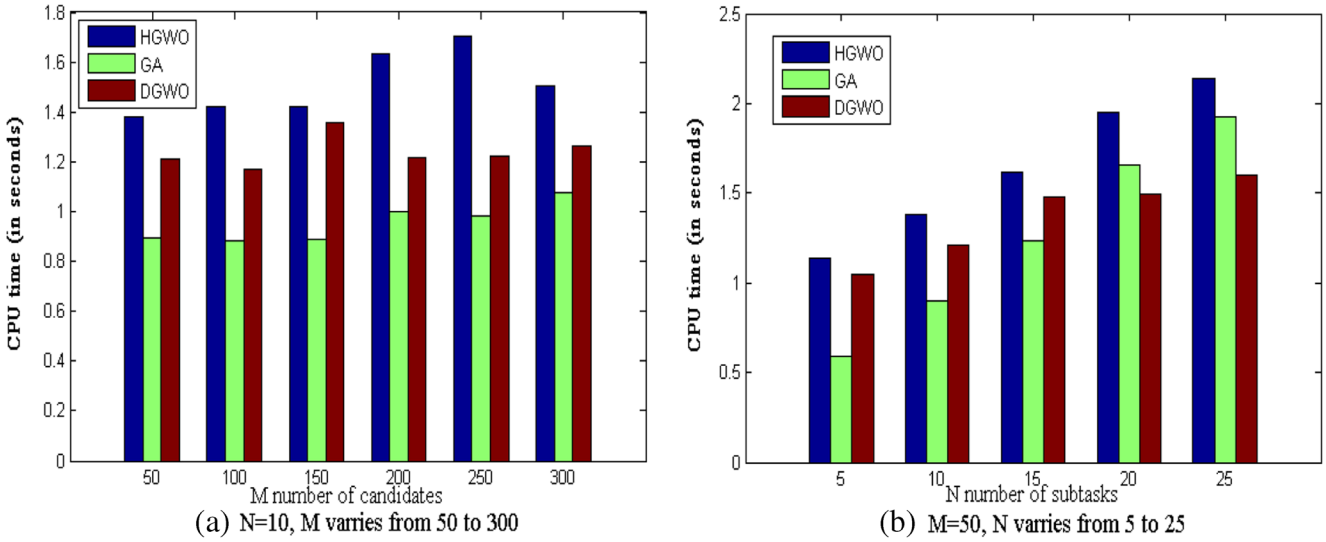

Fig. 5 Time consumption comparison for different scales. **a** $N = 10$, M varies from 50 to 300. **b** $M = 5$, N varies from 5 to 25

HGWO (this is the case expect for the last two cases of scenario 2, where $N = 20$ or $N = 25$). To clarify the reason behind this pattern, we have to take a look at the structure of these algorithms. In fact, the proposed HGWO algorithm takes advantage of both haunting process (the main time-consuming mechanism of DGWO) and crossover and mutation processes (the main evolutionary mechanisms of GA) and as a result, takes more time to complete a certain number of iterations compared with each of these algorithms. Nevertheless, because of the superior performance of the proposed HGWO algorithm and considering the fact that the higher CPU time is so insignificant even in the worst case scenario (2.13 s in the $N = 15$ and $M = 50$ case), HGWO would be chosen over GA and DGWO to solve the SCOS problem. Besides, the results show a similar pattern to the fitness values, meaning that computational time is more influenced by an increase in the number of subtasks compared with an increase in the number of candidates. This can again be contributed to the bigger impact of M on the size of solution space than N, and also the fact that computation of aggregated QoS values for a composite task with a higher number of subtasks, needs performing more arithmetic operations and consequently, consumes more time (see Table 1).

6.2 Effect of QoS ranges

In order to verify the effect of QoS value ranges on the performance of the proposed HGWO algorithm, 15 possible combinations of three different ranges including [0.5, 0.7], [0.7, 0.9] and [0.5, 0.9] are generated as listed in Table 3. In this experiment, the number of available candidates N is set to 200 and the number of subtasks M is set 10. Table 4 lists the comparison results of the optimal QoS fitness values obtained by each algorithm for each considered set of QoS values. As one can see from that table, HGWO outperforms the GA and DGWO in 11 cases out of the total of 15 sets (superiority rate of $11/15 = 73.3\%$). GA obtains the highest optimal fitness value in four of the cases (superiority rate of $4/15 = 26.7\%$) and DGWO performs the poorest among all experiments. In addition, DGWO has obtained an average value of 0.54453, followed by an average of 0.489879 for GA and 0.31932 for DGWO, which indicates the better overall performance of the proposed algorithm as the range of QoS values varies. Therefore, it can be concluded that the proposed HGWO algorithm is a better choice compared to the GA and DGWO, when the service composition problem has to deal with a varying range of QoS values.

Table 3 Sets of QoS ranges

Sets	Time	Cost	Reliability	Availability
1	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.7]
2	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.7]	[0.7, 0.9]
3	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.9]
4	[0.5, 0.7]	[0.5, 0.7]	[0.7, 0.9]	[0.7, 0.9]
5	[0.5, 0.7]	[0.5, 0.7]	[0.7, 0.9]	[0.5, 0.9]
6	[0.5, 0.7]	[0.5, 0.7]	[0.5, 0.9]	[0.5, 0.9]
7	[0.5, 0.7]	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]
8	[0.5, 0.7]	[0.7, 0.9]	[0.7, 0.9]	[0.5, 0.9]
9	[0.5, 0.7]	[0.7, 0.9]	[0.5, 0.9]	[0.5, 0.9]
10	[0.5, 0.7]	[0.5, 0.9]	[0.5, 0.9]	[0.5, 0.9]
11	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]
12	[0.7, 0.9]	[0.7, 0.9]	[0.7, 0.9]	[0.5, 0.9]
13	[0.7, 0.9]	[0.7, 0.9]	[0.5, 0.9]	[0.5, 0.9]
14	[0.7, 0.9]	[0.5, 0.9]	[0.5, 0.9]	[0.5, 0.9]
15	[0.5, 0.9]	[0.5, 0.9]	[0.5, 0.9]	[0.5, 0.9]

Table 4 Comparison results for each set of QoS ranges

Set	HGWO	GA	DGWO
1	*0.7246*	0.509364	0.377289
2	*0.55024*	0.496296	0.39358
3	*0.54024*	0.468342	0.352245
4	*0.55521*	0.42813	0.240125
5	*0.57765*	0.446643	0.197525
6	*0.5345*	0.454167	0.166672
7	0.48792	*0.580671*	0.499651
8	0.51453	*0.58941*	0.362515
9	*0.50895*	0.411912	0.337499
10	0.36951	*0.551457*	0.253033
11	*0.71412*	0.536931	0.445687
12	*0.7455*	0.53514	0.325672
13	*0.42242*	0.359397	0.329998
14	0.35261	*0.580788*	0.235945
15	*0.5705*	0.399888	0.272606
Average	*0.54453*	0.489879	0.31932
Superiority rate	*73.3%*	26.7%	0

The highest value of each row is indicated by an italic font.

6.3 Effect of QoS preferences

The corresponding weights of the four attributes comprising the QoS function were all set to 0.25 in the abovementioned simulations. However, this is not always the case with regard to customer preferences. As a result, in order to investigate the performance of the proposed HGWO algorithm when dealing with different QoS preferences, 10 different random QoS vectors are considered as shown in Table 5. The experiments are conducted assuming a SCOS problem consisting of 10 subtasks ($N = 10$) and 200 available candidate services for each subtask ($M = 200$), similar to the previous experiment in Section 5.1.1. The results are presented in Table 6. As one can observe from Table 6, in 8 cases (out of a total of 10 vectors), the optimal QoS value obtained by HGWO is higher than what is found by GA and DGWO. The average optimal fitness value obtained by HGWO equals to 0.6444, followed by 0.5672 for GA and 0.4471 for DGWO. In addition, as the weights associated with time and cost increase (or weights associated with reliability and availability decrease equivalently), the optimal QoS fitness value decreases drastically in all three algorithms. This happens in vector numbers 3, 4, 6, and 8 in which almost all three algorithms obtain their lowest QoS value. On the contrary, in vector numbers 1, 2, 7, and 10, with higher availability and reliability weights, the algorithms achieve relatively higher fitness values. This can be attributed to the fact that, in a sequential composition, availability, and reliability QoS values of the collaborating services will be multiplied together according to the third and fourth columns of Table 1, while the QoS values associated with time and cost will be added together (see the first and second columns of Table 1). As a result, an equal decrease in the QoS value of availability or reliability will lead to a higher drop in the value of aggregated QoS, as opposed to the amount of drop resulting from an equal decrease in time and cost Qos. In summary, the results prove the superior performance of HGWO compared to GA and DGWO when facing different user QoS preferences.

Table 5 Different user QoS preference vectors

Vectors	w_1	w_2	w_3	w_4
1	0.18	0.16	0.33	0.33
2	0.11	0.28	0.24	0.37
3	0.29	0.32	0.11	0.28
4	0.46	0.11	0.08	0.35
5	0.34	0.12	0.22	0.32
6	0.35	0.38	0.22	0.05
7	0.1	0.17	0.56	0.17
8	0.35	0.1	0.4	0.15
9	0.13	0.16	0.4	0.31
10	0.15	0.36	0.25	0.24

Table 6 Comparison results of different user QoS references

Vectors	HGA	GA	DGWO
1	*0.64417*	0.575928	0.459644
2	*0.78924*	0.575982	0.458107
3	0.54774	*0.557496*	0.478126
4	*0.55317*	0.552339	0.434594
5	*0.58366*	0.564813	0.441499
6	*0.61126*	0.606447	0.363826
7	*0.75292*	0.553266	0.501433
8	0.54141	*0.596376*	0.393712
9	*0.66592*	0.557757	0.47542
10	*0.7550*	0.532116	0.46469
Average	*0.644449*	0.567252	0.4471051
Superiority rate	*80%*	20%	0

The highest value of each row is indicated by an italic font.

7 Conclusions and recommended future work

Service composition and optimal selection is one of the most pivotal problems in the field of CMfg. The complexity associated with this problem arises from the fact that as the size of the solution space grows, the probability of searching that in polynomial time decreases. This very fact explains the growing interest dedicated to developing novel suitable algorithms that can deal with the problem more efficiently. For addressing such SCOS problems with large solution spaces, a hybrid grey wolf optimizer algorithm with evolutionary operators has

been proposed in this paper. In the structure of the proposed algorithm, the embedded crossover and mutation operators carry out a twofold functionality: they make it possible to adapt a continuous algorithm such as GWO to a combinatorial problem such as SCOS, and they help to avoid the stagnation at the hunting process through providing more exploration strength. Needless to say, the proposed structure of the HGWO algorithm can also be implemented to solve any other combinatorial problem without the need of significant further adaption and is expected to obtain competitive or superior results relative to existing metaheuristics.

In order to validate the efficacy of the proposed HGWO algorithm in solving large-scale SCOS problems, a series of experiments were designed and conducted which proved the superior performance of the algorithm compared to GA and DGWO algorithms. The results demonstrated that HGWO can obtain higher optimal fitness values in the majority of the experimental cases, at the expense of a negligible increase in computational time. It was also verified that the proposed algorithm remains superior when facing different QoS preferences and/or varying QoS value ranges.

In this work, we have converted the SCOS problem to a single objective optimization model using the SAW method. However, there are some drawbacks to this approach when compared to Pareto-based multi-objective methods [45]. Also, the services involved in service composition are treated as independent ones from each other, which can be further improved by considering correlations among them. People may extend the proposed algorithm with Pareto-based approaches to solve correlation-aware multi-objective SCOS problems, and also implement the HGWO to solve other combinatorial optimization problems.

References

1. Krishnaiyer K, Chen FF, Burgess B, Bouzary H (2018) D3S model for sustainable process excellence. Procedia Manufacturing 26: 1441–1447. https://doi.org/10.1016/j.promfg.2018.07.100
2. Liu N, Li X (2015) Granulation-based resource classification in cloud manufacturing. Proc Inst Mech Eng B J Eng Manuf 229(7): 1258–1270. https://doi.org/10.1177/0954405415572644
3. Zhang Y, Zhang G, Liu Y, Hu D (2017) Research on services encapsulation and virtualization access model of machine for cloud manufacturing. J Intell Manuf 28(5):1109–1123. https://doi.org/10.1007/s10845-015-1064-2
4. Wang T, Guo S, Lee C-G (2014) Manufacturing task semantic modeling and description in cloud manufacturing system. Int J Adv Manuf Technol 71(9):2017–2031. https://doi.org/10.1007/s00170-014-5607-z
5. Sahba R, Ebadi N, Jamshidi M, Rad P (2018) Automatic text summarization using customizable fuzzy features and attention on the context and vocabulary. In: 2018 World Automation Congress (WAC), 3–6 June 2018. pp 1–5. https://doi.org/10.23919/WAC.2018.8430483
6. Ostasevicius V, Jurenas V, Markevicius V, Gaidys R, Zilys M, Cepenas M, Kizauskiene L (2016) Self-powering wireless devices for cloud manufacturing applications. Int J Adv Manuf Technol 83(9):1937–1950. https://doi.org/10.1007/s00170-015-7617-x
7. Zhong RY, Lan S, Xu C, Dai Q, Huang GQ (2016) Visualization of RFID-enabled shopfloor logistics big data in cloud manufacturing. Int J Adv Manuf Technol 84(1):5–16. https://doi.org/10.1007/s00170-015-7702-1
8. Zarreh A, Saygin C, Wan H, Lee Y, Bracho A (2018) A game theory based cybersecurity assessment model for advanced manufacturing systems. Procedia Manufacturing 26:1255–1264. https://doi.org/10.1016/j.promfg.2018.07.162
9. Bracho A, Saygin C, Wan H, Lee Y, Zarreh A (2018) A simulation-based platform for assessing the impact of cyber-threats on smart manufacturing systems. Procedia Manufacturing 26:1116–1127. https://doi.org/10.1016/j.promfg.2018.07.148
10. Liang G, Shilong W, Ling K, Yang C (2015) Agent-based manufacturing service discovery method for cloud manufacturing. Int J Adv Manuf Technol 81(9–12):2167–2181. https://doi.org/10.1007/s00170-015-7221-0
11. Lv H, Xu Z (2016) Resource matching model of cloud manufacturing platform based on granularity optimization of the SFLA. Revista Tecnica de la Facultad de Ingenieria Universidad del Zulia 39(9):297–307. https://doi.org/10.21311/001.39.9.38
12. Xu Y, Chen G, Zheng J (2016) An integrated solution—KAGFM for mass customization in customer-oriented product design under cloud manufacturing environment. Int J Adv Manuf Technol 84(1): 85–101. https://doi.org/10.1007/s00170-015-8074-2
13. Kai Y, Ying C, Fei T (2016) A trust evaluation model towards cloud manufacturing. Int J Adv Manuf Technol 84(1–4):133–146. https://doi.org/10.1007/s00170-015-8002-5
14. kulj G, Vrabi R, Butala P, Sluga A (2017) Decentralised network architecture for cloud manufacturing. Int J Comput Integr Manuf 30(4–5):395–408. https://doi.org/10.1080/0951192X.2015.1066861
15. Ferreira L, Putnik G, CruzCunha MM, Putnik Z, Castro H, Alves C, Shah V, Varela L (2017) A cloud-based architecture with embedded pragmatics renderer for ubiquitous and cloud manufacturing. Int J Comput Integr Manuf 30(4–5):483–500. https://doi.org/10.1080/0951192X.2017.1291995
16. Lartigau J, Xu X, Nie L, Zhan D (2015) Cloud manufacturing service composition based on QoS with geo-perspective transportation using an improved Artificial Bee Colony optimisation algorithm. Int J Prod Res 53(14):4380–4404
17. Li H-F, Zhao L, Zhang B-H, Li J-Q Service matching and composition considering correlations among cloud services. In: Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on, 2015. IEEE, pp 509–514
18. Zheng H, Feng Y, Tan J (2016) A fuzzy QoS-aware resource service selection considering design preference in cloud manufacturing system. Int J Adv Manuf Technol 84(1–4):371–379
19. Wei X, Liu H (2015) A cloud manufacturing resource allocation model based on ant colony optimization algorithm. Int J Grid Distributed Comput 8(1):55–66
20. Zhou J, Yao X (2017) A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition. Int J Prod Res 55(16):4765–4784
21. Seghir F, Khababa A (2016) A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. J Intell Manuf 29:1773–1792. https://doi.org/10.1007/s10845-016-1215-0

22. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
23. Komaki GM, Kayvanfar V (2015) Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. J Comput Sci 8:109–120. https://doi.org/10.1016/j.jocs.2015.03.011
24. Sharma Y, Saikia LC (2015) Automatic generation control of a multi-area ST – thermal power system using Grey Wolf Optimizer algorithm based classical controllers. Int J Electr Power Energy Syst 73:853–862. https://doi.org/10.1016/j.ijepes.2015.06.005
25. Kamboj VK, Bath SK, Dhillon JS (2016) Solution of non-convex economic load dispatch problem using Grey Wolf Optimizer. Neural Comput & Applic 27(5):1301–1316. https://doi.org/10.1007/s00521-015-1934-8
26. Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. Neurocomputing 172:371–381. https://doi.org/10.1016/j.neucom.2015.06.083
27. Song X, Tang L, Zhao S, Zhang X, Li L, Huang J, Cai W (2015) Grey Wolf Optimizer for parameter estimation in surface waves. Soil Dyn Earthq Eng 75:147–157. https://doi.org/10.1016/j.soildyn.2015.04.004
28. Tao F, LaiLi Y, Xu L, Zhang L (2013) FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans Ind Inform 9(4):2023–2033. https://doi.org/10.1109/TII.2012.2232936
29. Xiang F, Jiang G, Xu L, Wang N (2016) The case-library method for service composition and optimal selection of big manufacturing data in cloud manufacturing system. Int J Adv Manuf Technol 84(1–4):59–70
30. Zhang W, Yang Y, Zhang S, Yu D, Xu Y (2016) A new manufacturing service selection and composition method using improved flower pollination algorithm. Math Probl Eng 2016:1–12. https://doi.org/10.1155/2016/7343794
31. Zhou J, Yao X (2017) Multi-objective hybrid artificial bee colony algorithm enhanced with Lévy flight and self-adaption for cloud manufacturing service composition. Appl Intell 1–22
32. Jin H, Yao X, Chen Y (2015) Correlation-aware QoS modeling and manufacturing cloud service composition. J Intell Manuf 28(8):1947–1960
33. Liu Y, Xu X, Zhang L, Tao F (2016) An extensible model for multitask-oriented service composition and scheduling in cloud manufacturing. J Comput Inf Sci Eng 16(4):041009
34. Liu W, Liu B, Sun D, Li Y, Ma G (2013) Study on multi-task oriented services composition and optimisation with the 'multi-composition for each Task'pattern in cloud manufacturing systems. Int J Comput Integr Manuf 26(8):786–805
35. Liu B, Zhang Z (2017) QoS-aware service composition for cloud manufacturing based on the optimal construction of synergistic elementary service groups. Int J Adv Manuf Technol 88(9–12):2757–2771
36. Bouzary H, Chen FF (2018) Service optimal selection and composition in cloud manufacturing: a comprehensive survey. Int J Adv Manuf Technol 97(1):795–808. https://doi.org/10.1007/s00170-018-1910-4
37. Bouzary H, Chen FF, Krishnaiyer K (2018) Service matching and selection in cloud manufacturing: a state-of-the-art review. Procedia Manufacturing 26:1128–1136. https://doi.org/10.1016/j.promfg.2018.07.149
38. Zhou J, Yao X (2017) A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition. Int J Adv Manuf Technol 88(9–12):3371–3387
39. Zhou J, Yao X (2017) Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing. Appl Soft Comput 56:379–397
40. Li L, Sun L, Guo J, Qi J, Xu B, Li S (2017) Modified discrete grey wolf optimizer algorithm for multilevel image thresholding. Comput Intell Neurosci 2017
41. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14. https://doi.org/10.1016/j.swevo.2012.09.002
42. Wang D, Yang Y, Mi Z (2015) A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electr Eng 43:129–141. https://doi.org/10.1016/j.compeleceng.2014.10.008
43. Wu Q, Zhu Q, Zhou M (2014) A correlation-driven optimal service selection approach for virtual enterprise establishment. J Intell Manuf 25(6):1441–1453. https://doi.org/10.1007/s10845-013-0751-0
44. Tao F, Zhao D, Hu Y, Zhou Z (2008) Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Trans Ind Inform 4(4):315–327. https://doi.org/10.1109/TII.2008.2009533
45. Chiandussi G, Codegone M, Ferrero S, Varesio FE (2012) Comparison of multi-objective optimization methodologies for engineering applications. Comput Math Appl 63(5):912–942. https://doi.org/10.1016/j.camwa.2011.11.057