IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 36, No. 3, June, 2022, pp. 417-425
Journal homepage: http://iieta.org/journals/ria

DDAPSO: Hybrid Discrete Dragonfly Algorithm and Particle Swarm Algorithm to Service Selection and Composition for the Internet of Things Applications

Bilal Benmessahel[1*], Farid Nouioua[2,3]

[1] Mechatronics Laboratory (LMETR) - E1764200 and Faculty of Technology, Setif 1 University, Setif 19000, Algeria
[2] LMSE and Computer Science Department, University of Bordj Bou Arreridj, Bordj Bou Arreridj 34000, Algeria
[3] LIS UMR – CNRS 7020, Aix-Marseille University, Campus de St-Jérôme, Marseille 13397, France

Corresponding Author Email: bilal.benmessahel@gmail.com

https://doi.org/10.18280/ria.360309

Received: 16 February 2022
Accepted: 8 June 2022

Keywords:
discrete dragonfly algorithm, particle swarm algorithm, service selection, Internet of Things (IoT)

ABSTRACT

Recently, the Internet of Things (IoT) has quickly risen as one of the most essential technologies of this century. IoT allows users to connect to a vast network of smart devices, services, and data. An important and challenging research problem in the Internet of Things applications is how to select an appropriate service selection (SS). In the SS problem, users can combine several services from diverse sources (things or devices) to satisfy their needs. On the other hand, the SS problem is known for its complexity and is categorized as an NP-hard problem; such problems are typically solved utilizing heuristics like bio-inspired algorithms. In this research a new bio-inspired algorithm called DDAPSO is created to solve the SS problem where a new strategy is proposed to maintain a balance between the exploration and exploitation abilities. This hybrid algorithm is the result of coupling a Discrete Dragonfly Algorithm (DDA) with the particle swarm optimization algorithm (PSO). The suggested algorithm was properly tested using a variety of scenarios with different numbers of services and with different numbers of concrete services per each service set or task. The proposed algorithm is compared with the main recent well-known algorithms, i.e. GA, PSO, DDA, ABC and MVO for service selection. In a large-scale setting, the results clearly show that the DDAPSO algorithm outperforms other services selection algorithms reported in the literature in terms of selection optimally as well as execution time.

1. INTRODUCTION

Nowadays, Internet of Things (IoT) is a hot spot for researchers and industries. It establishes the road for the development of new intelligent applications that may be used to provide unique and important services in a variety of fields, including smart cities/homes, transportation, healthcare monitoring, industrial automation, agriculture, etc. [1].

IoT is composed of many smart objects providing software services that abstract diverse functionalities (abstract service (AS)). Composite services (CS) are made up of these functionalities that can be combined to develop new applications, to meet complicated user needs and specific quality of service (QoS) requirements (such as response speed, throughput, availability, price, popularity, etc.).

Service-oriented architecture (SOA) encourages the development of complex applications by combining atomic services to deliver new functions that none of the services could give individually [2].

The word "concrete service" refers to a real service in this work, whereas an abstract service, also known as a class of services or tasks, defines the concrete service abstractly. There may be numerous physical services for each abstract service, each with the same functionality, but maybe varying in quality levels.

The composition method consists in generating a new service class by putting together existing classes in a plan or actions flow, and then determining the best bindings between these classes and their concrete services, also known as candidate services [3].

The QoS-aware service selection and composition (QoS-SC) problem is a combinatorial optimization problem that is also an NP-hard problem. Metaheuristics, such as bio-inspired algorithms, are commonly used to solve such types of problems. The goal of this research is to create a new hybrid bio-inspired algorithm "DDAPSO" to solve the QoS-SC problem. In DDAPSO two algorithms are coupled: The first one is a Discrete Dragonfly Algorithm (DDA) and the second one is a particle swarm optimization algorithm (PSO).

To conduct the evolutionary process, The proposed algorithm combines two phases: The first one is the exploration phase and uses a discrete version of the Dragonfly Algorithm (DDA) and the second one is the exploitation phase, which is considered as a local search strategy, and uses the particle swarm optimization algorithm (PSO). The exploitation phase improves on the findings of the exploration phase by looking deeper into the process of finding potential places in search space.

When there is no nearby dragonfly, the discrete dragonfly algorithm [4] uses the Levy flight technique to enhance random behavior in the exploitation phase. This could considerably improve the algorithm's exploring process. However, the best personal solution of dragonflies is not used in the process. Due to this, DDA converges to the optimal

solution very slowly and can become stuck in the local optima or perhaps be unable to find an optimal solution.

On another side, particle swarm optimization (PSO) [5] has been demonstrated in many prior works to discover the best solutions to a variety of problems. Since may rapidly converge to the optimal solution, it is a good candidate for the exploitation phase because it equated finding the optimal solution by exploiting the best solutions of the particles.

The subsequent parts of the paper are organized as follows. Section 2 introduces the related work. The QoS aware service in IoT and the Mathematical model for QSS are presented in section 3. In section 4, the proposed algorithm is presented in detail, including the solution schema and its fitness evaluation and the two phrases used in the algorithm. Section 5 analyses and discusses the experimental results. Finally, conclusions and future work are given in Section 6.

2. RELATED WORK

In IoT many heterogeneous smart objects are connected, each object can provide a service. To develop and commercialize more complicated IoT applications with advanced capabilities, smart object services must be integrated and composed. In the literature, many approaches for selecting and composing IoT services based on their function have been developed [6, 7]. This section gives a brief review of some related works and approaches used to solve the service selection and composition problem in IoT.

In Ref. [8], to locate the candidate service with the best QoS, the authors used a lexicographic optimization strategy and a quality of service constraint relaxation technique. Then, using a simple weighting method, the IoT service selection problem is turned into a mono-objective optimization problem, and the final selected service that meets the user's QoS needs is derived from the candidate service.

To improve the quality of service factors in cloud-edge computing, Hosseinzadeh et al. [9] propose a hybrid Artificial Neural Network-based Particle Swarm Optimization (ANN-PSO) Algorithm. A meta-heuristic and machine learning algorithm for evaluating service selection problems is proved in which formal approaches are used to ensure that functional and non-functional specifications are met. Also, they provide a formal verification method for checking some essential Linear Temporal Logics (LTL) formulations based on a labelled transition system.

Alizadeh et al. [10] give a method to choose the best composition out of all feasible combinations without knowing a priori the preferences quality of service of the users. For determining the best QoS-aware service composition, they offer a vector-valued MDP technique. The method solves MDP using dynamic programming and learns the user's preferences through direct queries.

According to Khanouche et al. [11], to obtain a solution which is very close to the optimal composition in an acceptable amount of time, an improved teaching learning-based QoS-aware services composition algorithm (ITL-QCA) is suggested. The proposed services composition algorithm is distinguished by a small number of tuning parameters and a high exploration capability of the composition search space. This enables the creation of compositions with high QoS optimality without the need for any hard tuning parameters.

Also, Khanouche et al. [12] proposes an energy-centered and QoS-aware services selection algorithm (EQSA) for IoT services selection. Using a lexicographic optimization strategy and a QoS constraints relaxation technique, the suggested selection approach consists of pre-selecting the services that provide the QoS level required for user satisfaction.

Kurdi et al. [13] propose in a multi-cloud environment, a bio-inspired algorithm that simulates the behavior of cuckoo birds and is used to identify a composite service that fulfills a user's request. They present a problem-dependent heuristic that takes into account the SC problem and its unique characteristics in a multi-cloud setting.

In order to establish an ideal balance between QoS level and consumed energy of IoT service composition, Alsaryrah et al. [14] model the problem as a bi-objective shortest path optimization (BSPO) problem which has been solved using an exact technique called pulse.

Li et al. [15] formulate the QoS-oriented service composition problem as a multi-criteria goal programming (MCGP) problem, and the obtained model is solved using a multi-population genetic algorithm (MGA). MCGP not only finds non-inferior composite services by lowering QoS limitations to meet users' QoS requirements, but it also allocates high-quality Web services to combine a composite service.

Kashyap et al. [16] use the NSGA2 multi-objective metaheuristic search method to find the best solution to the Service Composition Problem (SCP) in the context of Internet of Things.

In summary, researchers seek to identify the best services selection for composing an ideal composite service in IoT applications. The development of efficient procedures remains a critical endeavour. In reality, bio-inspired techniques can produce near-optimal results in a short amount of time. However, there are several limitations to this type of technique, such as premature or local optimum stagnation and sluggish convergence to reach near-optimal solutions. Furthermore, because the QoS-vast CSC's search space is a significant challenge in terms of solution accuracy, our objective in this study will be to propose a new effective algorithm to address the problem.

3. QOS AWARE SERVICE SELECTION IN IOT

As stated in the introduction, QoS aware service selection and composition in IOT refers to the combination of relevant abstract services given by multiple service providers clustered together in order to serve the user request. Different services are required for the execution of these requests by the user. Workflow refers to the sequence of specialized services required to complete a task. Figure 1 shows the general schema of a workflow selection.

An example taken from [17] is about trip planning in the context of smart cities. A trip-planning application allows users to automatically design a trip to visit places of interest (PoIs) in the city using the IoT infrastructure. In this example the compound task (trip-planning) uses different primitive tasks or sub-services for booking local transportation based on time preference, weather conditions, person's current location, nearest available cab, route-planning service, visiting hours of PoIs around the city and automatic payment service. For each of the sub-services evoked above, we can find multiple concretes services providers using IoT devices. The difference between these similar concrete services lies in their quality of services.

The availability of functionally similar services increases the complexity of selecting the best concrete services. So, our aim in this work is to propose an efficient method that can perform the selection of the best concrete services in terms of quality of service in order to satisfy the requirements of the end user.

Besides, the service selection and composition in the IoT architecture is depicted in Figure 2. The service composition of the IoT architecture is divided into five layers, the relationship between the five layers is that each layer receives as input the output of the previous layer, enriches it and feeds the following layer. This relationship is similar to that found between ISO layers in computer networks. The roles of each layer are given as follows:

- The perception layer which is composed of sensors and smart devices which are responsible of providing the service as data or processing resource.
- The service provided with perception layer is send to network layer which is responsible of transferring the service to the cloud layer.
- The cloud layer presents the database of all services and provides the different services by public or private cloud.
- The service selection and composition layer is responsible of selecting the best sub services and composing this later to achieve the global task.
- The application layer presents the interface where the end users can use the service or requests the service.

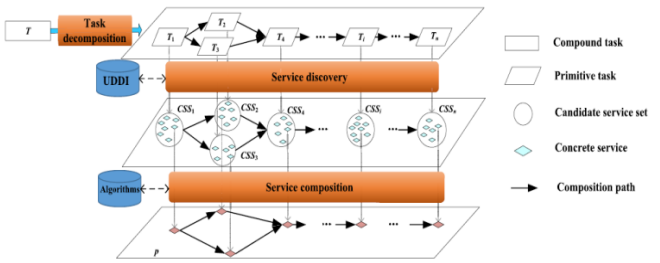

Figure 1. QoS aware service selection and composition [18]

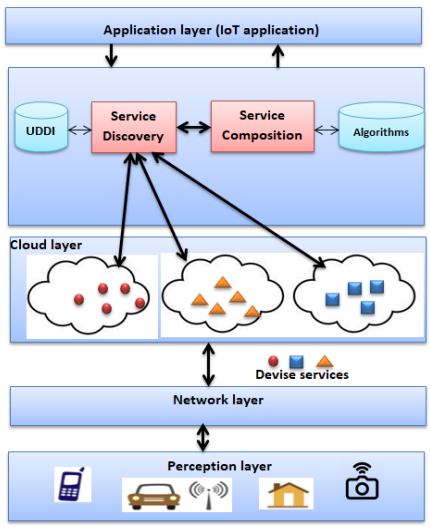

Figure 2. QoS aware service composition in IOT

3.1 QoS model for service selection and composition

We identify seven QoS characteristics as the quality evaluation criteria of service according to domain application of IoT, based on various QoS attributes for services published by W3C working group.

(1) Execution time. The average time between when the user sends the request and when the server responds is the execution time of a service.

(2) Reliability. The percentage of service requests that are executed successfully is the reliability of a service. The ratio of successful executions to total number of services called is used to calculate it.

(3) Execution cost. is the price to pay (in terms of money) in order to exploit the service.

(4) Availability. During a certain time interval, the percentage of time that a service is available.

(5) Scalability. The IoT environment's ability to be modified and changed in many conditions.

(6) Reputation. is a measure of its trustworthiness. It mainly depends on end user's experiences of using the services. A level of satisfaction for this QoS can be defined as "very high", "high", "normal", "poor" or "very poor". We can just use a ranking technique based on end user's experiences of using the services for calculating the reputation quality.

(7) Response Time. It's the amount of time between when a user asks a service and when they get an answer.

The most extensively used service composition workflows are: (a) sequential, (b) Loop, (c) Parallel and (d) Switch as shown in Figure 3. The mathematical expressions used to calculate the value of each QoS for the different workflow are given in Table 1.

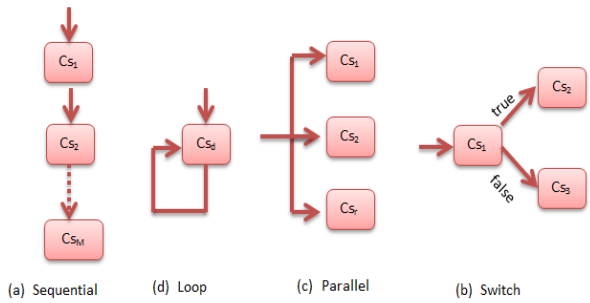

Figure 3. Workflows used in service composition

3.2 Mathematical model for QSC

In a service selection and compositions model, tasks are represented as $T=t_1, t_2, t_3, ..., t_n$ and the workflow must be selected (sequential, Loop, Parallel and Switch).

Suppose that we have m atomic services given as $S=s_1, s_2, s_3, ..., s_m$ with the same functionality but varying only in QoS. For each atomic service s_i, the QoS is expressed as t_i time, a_i availability, r_i reliability, and c_i cost. As a result, there are a lot of possibilities to allocate an atomic service from m services for n tasks.

The service composition approach should also satisfy the service level agreement (SLA), which is the level of service required from the service provider. SLA is a contract between a service provider and a consumer that ensures a minimum level of service is maintained. It guarantees levels of reliability, availability and responsiveness to systems and applications.

In other words, the service selection and composition algorithm's input is:

- The task list $T=t_1, t_2, t_3, ..., t_n$;
- Their workflow;
- The set of atomic services $S=s_1, s_2, s_3, ..., s_m$ for each task;

• The QoS for each atomic service s_i, i=1, ..., m as {t_i, a_i, r_i, c_i};

• SLA vector SL=(SLt, SLa, SLr, SLc) for QoS attribute requested by the user.

The results are the chosen atomic service for each job from a set of m atomic services that optimize QoS while meeting the SLA requirements.

We evaluate the fitness value of the service composition solution based on the QoS values of its selected atomic service for each task, as well as composite models (i.e. sequential, Loop, Parallel, and Switch) that specify the interconnection architecture between these atomic services (see Eq. (3) below).

Cost and benefit criteria can be used to categorize the characteristics of QoS. A larger value of a cost criterion corresponds to a lower quality (e.g. execution price or execution time). In contrast, a higher value of a benefit criterion corresponds to a higher quality (e.g., reliability and availability). The following relationship can be used to normalize QoS attributes:

• normalization of aggregate cost attributes (i.e. $q_i \in$ cost criteria):

$$Q(q_i) = \begin{cases} \dfrac{agg(max(q_i)) - agg(q_i)}{agg(max(q_i)) - agg(min(q_i))} \\ 1, if agg(max(q_i)) = agg(min(q_i)) \end{cases} \quad (1)$$

• normalization of aggregate benefit attributes (i.e. $q_i \in$ benefit criteria):

$$Q(q_i) = \begin{cases} \dfrac{agg(q_i) - agg(min(q_i)) -}{agg(max(q_i)) - agg(min(q_i))} \\ 1, if agg(max(q_i)) = agg(min(q_i)) \end{cases} \quad (2)$$

In Eqns. (1) and (2):

• $max(q_i)$ and $min(q_i)$ denote the maximum and minimum possible values of the i^{th} QoS criterion for the selected service compositions, respectively,

• $agg(max(q_t))$ denotes the aggregated value of the i^{th} QoS criterion of selected service compositions.

The set of possible aggregation operations is: agg={sum, $prod$, avg, max and min}. Each QoS attribute can be applied based on the composite model and the relevant QoS parameter's characteristic. For the sequential composite model, the sum operation (i.e. sum) is the aggregation operation of the QoS response time. We choose the sequential pattern in this study because the other three patterns, as described, can be easily transformed into the sequential pattern. The different aggregation operations for composition models are shown in Table 1 (sequential, Loop, Parallel and Switch).

If the user's preference for the i^{th} QoS attribute is $w_i \in [0,1]$ such that $\Sigma_{i=1}^{r} w_i = 1$, then the fitness value can be constrained by the following formula for r attributes of the composite service that satisfied users:

$$Fitness = \Sigma_{i=1}^{r} Q(q_i) * w_i \quad (3)$$

Table 1 show the different aggregation functions for QoS properties based on the different workflows used in this work.

In Table 1, t, c, a and r are the QoS attributes, n refers to the task number, j corresponds to j^{th} candidate service, and i corresponds to the i^{th} subtask. P_i stands for the conditional probability of the i^{th} task, and L is the iteration number.

Table 1. Aggregation functions for QoS properties based on the different workflows

QoS attribute	workflow			
	Sequential	Switch	Parallel	Loop
Response time	$\Sigma_{i=1}^{n} t_{r,i}^{j}$	$\Sigma_{i=1}^{n} t_{r,i}^{j}$	$\min_{1 \le i \le n} t_{r,i}^{j}$	$L * t_{r,i}^{j}$
Execution cost	$\Sigma_{i=1}^{n} c_{r,i}^{j}$	$\Sigma_{i=1}^{n} c_{r,i}^{j}$	$\min_{1 \le i \le n} c_{r,i}^{j}$	$L * c_{r,i}^{j}$
Availability	$\Pi_{i=1}^{n} a_{r,i}^{j}$	$\Pi_{i=1}^{n} a_{r,i}^{j}$	$\max_{1 \le i \le n} a_{r,i}^{j}$	$(a_{r,i}^{j})^{L}$
Reliability	$\Pi_{i=1}^{n} r_{r,i}^{j}$	$\Pi_{i=1}^{n} r_{r,i}^{j}$	$\max_{1 \le i \le n} r_{r,i}^{j}$	$(r_{r,i}^{j})^{L}$

4. THE PROPOSED APPROACH

In optimization, many algorithms suffer from the phenomenon of being locked in the local optima while the algorithms aim at finding the optimum solution to the problem. Any optimization algorithm is composed of the two main phases: The exploration phase and the exploitation phase. The primary goal of the exploration phase is to investigate the global space as widely as possible. Whereas, the exploitation phase builds on the findings of the exploration phase by delving deeper into the process of finding potential places in space.

The Dragonfly Algorithm (DA) [4], is a new optimization algorithm that is inspired by the behaviors of dragonflies in nature. It has been shown to be more effective and superior to various well-known meta-heuristics in the literature. DA is proposed for continuous functions and engineering problems. To make this algorithm work in the discrete space, this paper introduces a new version of DA called Discrete Dragonfly Algorithm (DDA). In DDA when there is no surrounding dragonfly, the Dragonfly algorithm uses the Levy flying to improve randomization and stochastic behavior. This could considerably improve the algorithm's exploring process. During the operation, however, the best experience, which is the personal best, of dragonfly is not used. The DDA converges to the optimal solution relatively slowly and as a result of this, it can become stranded in the local optima. On another side, Particle Swarm Optimization (PSO) [5] has been demonstrated in a number of works in the literature to discover the best answer to a variety of problems. PSO rapidly converge to the solution thanks to its equations that enable it to find the best solution by exploiting the best experience of the particles, i.e., it is a good algorithm at the exploitation level.

To take advantage of both algorithms, we develop in this paper a new method that combines the significant features of the DDA and PSO algorithms. the idea of the proposed method is to exploit DDA at the exploration phase and PSO at the exploitation phase. The dragonflies in DA are initially set to traverse the search space in order to locate the global solution region. At the end of the DDA phase, which happens when it cannot further improve its best solution, the best service selection (the best solution) found by DDA is used as the Global best solution (Gbest) of the PSO phase. So the transferred data between the two phases is the individual representing the best solution found by DDA.

Notice that in the proposed algorithm DDAPSO, we use a stagnation factor (SF). This factor is a counter which allows the passage from the exploration phase achieved by DDA to the exploitation phase performed by PSO. The implementation of SF is given in algorithm 1 below.

4.1 The exploration phase (DDA PHASE)

In this phase, we use DDA algorithm to locate the area where it is most likely to find the best solution for our problem. The following the mathematical formalization of the explanation task performed by DDA.

Suppose that we have a population of *N* dragonflies. Eq. (4) gives the position of the i[th] dragonfly.

$$X_i = \{x_i^1, x_i^2, \ldots, x_i^N\} \tag{4}$$

The solution in the search space corresponds to each dragonfly in the swarm. Separation, Alignment, Cohesion, Attraction towards food sources, and diversion towards enemy sources are five different operators that influence dragonfly swarm movement (see Figure 4).

The following is a mathematical model of each of these behaviors:

The below formula is used to compute the separation:

$$S_i = -\Sigma_{j=1}^{N} X - X_j \tag{5}$$

where, X represents the current individual's position, X_j represents the position of the j^{th} neighboring individual, and N represents the total number of surrounding members.

The following formula is used to calculate alignment:

$$A_i = \frac{\Sigma_{j=1}^{N} V_j}{N} \tag{6}$$

where, X_j denotes the j^{th} neighboring individual's velocity.

The following formula is used to calculate cohesion:

$$C_i = \frac{\Sigma_{j=1}^{N} X_j}{N} - X \tag{7}$$

The following formula is used to calculate the attraction to a food source:

$$F_i = X^+ - X \tag{8}$$

where, the position of the food source is represented by X^+.

The formula below is used to calculate enemy distraction:

$$E_i = X^- + X \tag{9}$$

where, the enemy's position is indicated by X^-.

A step vector should be calculated as follows to update the position of artificial dragonflies in a search space:

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_t \tag{10}$$

where, s denotes the weight of separation, and S_i denotes the i^{th} individual's separation, a denotes the alignment weight, A denotes the i^{th} individual's alignment, and c denotes the cohesion weight. C_i is the i^{th} individual's cohesiveness, f is the food factor, and the i^{th} individual's food supply is f_i. e is the enemy component, E_i is the i^{th} individual's adversary location, w is the inertia weight and t is the number of iterations.

The position vectors are calculated after the step vector has been determined:

$$X_{t+1} = X_t + \Delta X_t \tag{11}$$

When there are no neighboring solutions, the artificial dragonflies must fly across the search space utilizing a random walk (Levy flight) to improve their randomness, stochastic behavior, and exploration. In fact, The DDA phase use neighboring solutions to fly across the search space.

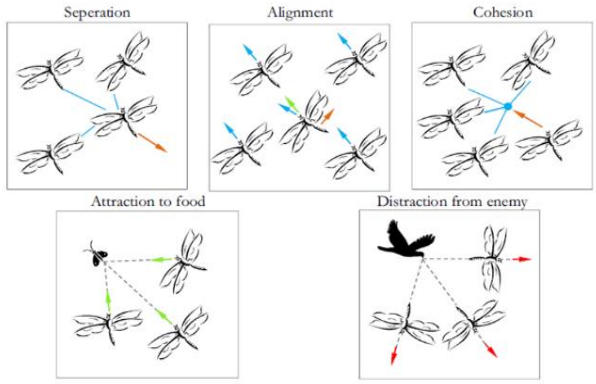

Figure 4. Different behaviours of Dragonflies in a swarm [4]

The random walk ((Levy flight) is used intermittently only when there are no improvement of the solution given by neighboring dragonflies. In this case, the objective of Levy flight technique is to try to bring out the DDA algorithm from the stagnation by exploring new regions of the search space.

The following equation is used to update the position of dragonflies in this case:

$$X_{t+1} = X_t + Levy(d)X_t \tag{12}$$

DDA uses an approximation function to convert form continues to discrete values as in Table 2.

Table 2. A solution for n=5 tasks and m=100 atomic services

	T_1	T_2	T_3	T_4	T_5
$T_i^j \in [0.5,100]$	10.6	3.7	52.4	86.8	8.1
$DT_i^j = round(T_i^j)$	11	4	52	87	8

4.2 The exploitation phase (PSO phase)

In this phase, we use discrete particle swarm optimization (DPSO) as exploitation process to exploit the results found by DDA in the exploration phase. Particle swarm optimization (PSO) is a population-based search strategy inspired by bird swarm behavior (information exchange) [5]. In PSO, a random population of particles is created at the beginning and these particles move at a given velocity based on their interactions with other particles in the population. The personal best of each particle, as well as the global best of all the particles, are tracked at each iteration, and the velocity of all the particles is modified depending on this information. Weights are assigned to the global and personal bests based on a set of parameters. To transform a continuous value to a discrete value, we employ an approximation function, just like in DDA. The notations used in this section are mostly based on the study [5]. Each particle is represented by a d dimensional vector that is randomly initialized with discrete values for each individual value.

$$X_i = (x_{i1}, x_{i2}, x_{i3}, \cdots, x_{id}) \tag{13}$$

The velocity is initialized to zero and represented as a d dimensional vector.

$$V_i = (v_{i1}, v_{i2}, v_{i3}, \cdots, v_{id}) \quad (14)$$

The best personal position that each particle has recorded is saved as:

$$P_i = (p_{i1}, p_{i2}, p_{i3}, \cdots, p_{id}) \quad (15)$$

Each particle adjusts its position according to its personal best (Pbest) and the global best (gbest) at each iteration.

$$V_i^{t+1} = W * V_i^t + c_1 * r_{i1} * (Pbest_i(t) - X_i^t) + c_2 * r_{i2} * (gbest_i(t) - X_i^t) \quad (16)$$

The acceleration constants c_1 and c_2 are referred to as cognitive and social parameters, respectively. The values r_1 and r_2 are random real values in the interval [0, 1]. The inertia weight is denoted by w. It regulates how the particle's past velocity affects the velocity in the next iteration.

4.3 Solution encoding

In the proposed DDAPSO, to represent each solution of QoS-oriented service selection and composition in IoT, we use an array of integer numbers. Each number in the array represents a specific concrete service from the task in question. Figure 5 shows a composite concrete IoT service consisting of five tasks, with each element in the array encoding the concrete IoT service chosen from its associated task. The third entry in the array, for example, indicates the concrete service number 25 of the third task.

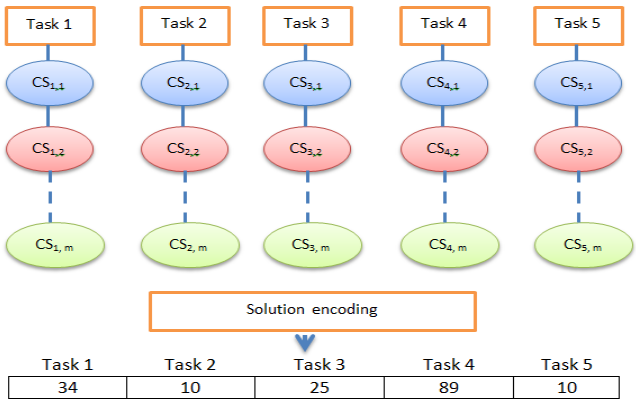

Figure 5. Solution encoding

4.4 The DDAPSO algorithm

The concepts issued from the discrete dragonfly and the discrete particle swarm optimization algorithms that are described in the previous sections are combined to obtain the DDAPSO algorithm that can benefit from their coexistence. The dragonfly algorithm has the ability to obtain diverse solutions with its formation of static swarms and the discrete PSO converging to the global best solution creates synergy in the implementation of the hybrid algorithm which results in increased performance. The Flowchart of DDAPSO is given in Figure 6. The Pseudo-codes of DDAPSO is shown in Algorithm 1.

Algorithm 1 Pseudo-codes of the DDAPSO algorithm

```
Initialize the dragonflies population Xi(i = 1, 2, ..., n)
Initialize step vectors δXi(i = 1, 2, ..., n)
Initialize stagnation factor = 0
while stagnation is not met do
    Calculate the objective values of all dragonflies
    if fitness(i) > bestfitness then
        bestfitness=fitness(i); bestposition=position(i,:);
        stagnation factor = 0;
    else
        stagnation factor += 1;
    end if
    Update the food source and enemy
    Update w, s, a, c, f, and e
    Calculate S, A, C, F, and E using Eqs. 5 to 9
    Update neighbouring radius
    if a dragonfly has at least one neighbouring dragonfly
then
        Update velocity vector using Eq.10
        Update position vector using Eq. 11
    else
        Update position vector using Eq. 12
    end if
end while
Check and correct the new positions based on the bound-
aries of variables
Initialize the particle's position
Initialize pBest to its initial position
Initialize gBest with the best solution found by DDA
Initialize velocity
while the end condition is not satisfied do
    Calculate the objective values of all particules
    if if current position is better than pBest then
        Update pBest
        assign pbest to gBest
        computes velocity
        update particules positions
    end if
end while
Output GBest that holds the best found solution.
```

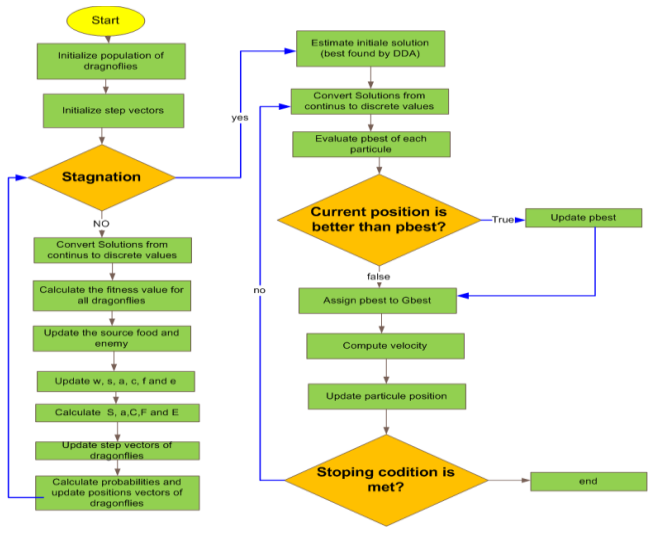

Figure 6. Flowchart of DDAPSO

5. EVALUATION METHODOLOGY

The following materials, which include both hardware and software components, have been used to implement and analyze the Hybrid Discrete Dragonfly Algorithm and Particle Swarm Algorithm:

- Hardware: Windows OS with 2.9 GHz Intel core i3 processor and 8 GB of RAM.
- Software: Matlab version 18a.

To verify the effectiveness, efficiency, and superiority of DDAPSO algorithm, its performance is compared with five recent optimization algorithms using large-scale intances of service selection and composition problem. These optimization algorithms are: Multi-verse optimization algorithm (MVO) [19, 20], Genetic algorithm (GA) [21, 22], Discrete version of Dragonfly Algorithm (DDA) [4], Particle Swarm Optimization Algorithm (PSO) [5] and Artificial Bee Colony Optimization Algorithm (ABC) [23]. All algorithms' parameter settings are shown in Table 3.

The above-mentioned references of the comparisons algorithms are used to choose these settings. Hence, these sources should be consulted for more information on the meaning and function of the parameters.

The population size for all algorithms was unified, the execution of all algorithms was performed 30 times, and each execution included 1000 iterations. Each repetition's average fitness value, performance stability, and execution time has been correctly recorded.

Table 3. Parameters settings for different Algorithms

Algorithms	Parameter Values: Pop size		
MVO	30	Wep max Wep min	1 0.3
GA	30	Pc PM	0.6 0.01
DDA	30		
PSO	30	Wmax Wmin C1 C2	0.9 0.2 2 2
ABC	30	L NG	1 3
DDAPSO	30	Wmax Wmin C1 C2	0.9 0.2 2 2

The experimental studies have been conducted using real datasets (Ver. 2.0) [24], which include 25 service selection and composition challenges. The datasets [24] contain about 2500 real tasks, each with nine QoS criteria. Cost, response time, availability, and dependability are the most effective QoS attributes that optimization algorithms use to solve the service selection and composition problem. With a total of 25 datasets, different scenarios were built with a number of tasks equal to 20, 40, 60, 80, and 100, and each task is composed of a number of concrete services equal to 200, 400, 600, 800, and 1000 as in Ref. [20].

Figures 7-10 represent the results of all algorithms in terms of the average solution where the scenarios have a workflow comprising 20, 40, 60, 80 and 100 abstract tasks and, each abstract task, comprising 200, 400, 600, 800, and 1000 concrete services. The average execution time is shown in Figure 9 for all algorithms.

We can observe from Figures 7 and 8 that for the average of the QoS fitness value, the proposed algorithm DDAPSO outperforms all the other algorithms for all scenarios. From Figure 8, where the number of concrete services is equal to 1000, the average of QoS fitness value obtained by DDAPSO is 5,27E-01, whereas that obtained by MVO, GA, DDA, PSO and ABC are 4,66E-01, 4,91E-01, 4,92E-01, 4,38E-01, 4,53E-01, respectively.

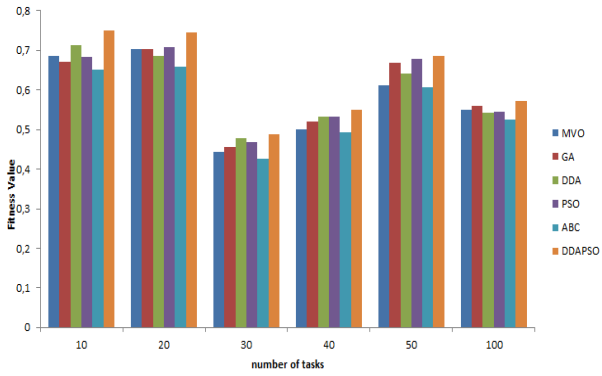

Figure 7. Comparison of results (scenario 1: the number of tasks *n* changes from 10 to 100, and the number of concrete services m varies from 10 to 100)

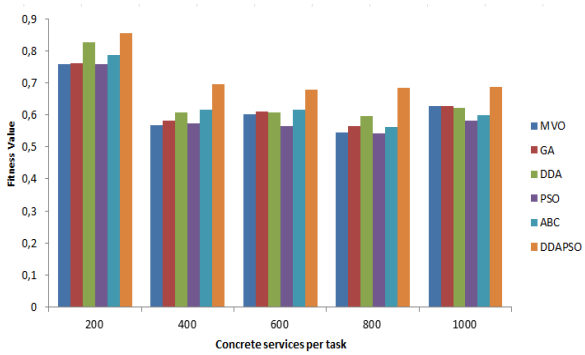

Figure 8. Comparison of results (scenario 2: the number of task *n*=20 and the number of concrete services *m* changes from 200 to 1000)

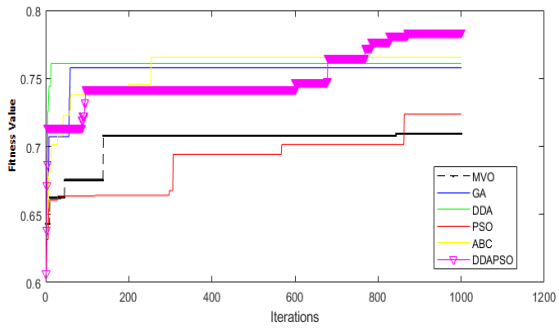

Figure 9. The quality of the optimal composite service obtained by the compared algorithms

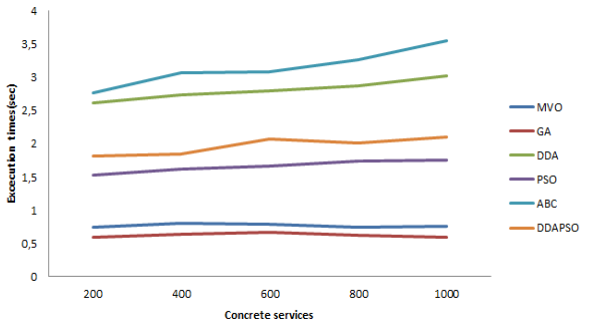

Figure 10. Comparison of computation times (scenario 3: the number of tasks *n*=20 and the number of concrete services m varies from 200 to 1000)

Table 4. Statistical comparison between DDAPSO and all other algorithms in terms of best solution

Average of Qos fitness Values						
Datasets	MVO	GA	DDA	PSO	ABC	DDAPSO
20-200	5,99E-01	5,91E-01	6,23E-01	5,86E-01	6,12E-01	**6,37E-01**
20-400	4,46E-01	4,70E-01	4,72E-01	4,62E-01	4,73E-01	**5,01E-01**
20-600	7,05E-01	7,33E-01	7,58E-01	6,96E-01	7,12E-01	**7,88E-01**
20-800	4,67E-01	5,13E-01	5,27E-01	4,95E-01	5,29E-01	**5,37E-01**
20-1000	5,84E-01	5,62E-01	6,16E-01	5,68E-01	5,68E-01	**6,41E-01**
40-200	7,15E-01	7,41E-01	7,65E-01	7,02E-01	7,36E-01	**7,88E-01**
40-400	5,22E-01	5,36E-01	5,80E-01	4,96E-01	5,37E-01	**5,94E-01**
40-600	5,19E-01	5,46E-01	5,58E-01	5,02E-01	5,22E-01	**6,06E-01**
40-800	3,93E-01	4,39E-01	4,56E-01	3,90E-01	4,21E-01	**4,57E-01**
40-1000	5,36E-01	5,53E-01	5,76E-01	5,27E-01	5,07E-01	**5,96E-01**
60-200	5,10E-01	5,54E-01	6,06E-01	4,97E-01	5,39E-01	**6,34E-01**
60-400	6,06E-01	6,44E-01	6,45E-01	5,88E-01	6,14E-01	**6,87E-01**
60-600	4,99E-01	5,31E-01	5,58E-01	4,84E-01	5,36E-01	**5,59E-01**
60-800	5,26E-01	5,57E-01	5,60E-01	4,94E-01	5,08E-01	**5,68E-01**
60-1000	5,45E-01	6,11E-01	5,99E-01	5,53E-01	5,68E-01	**6,47E-01**
80-200	6,24E-01	5,85E-01	5,64E-01	5,42E-01	5,60E-01	**6,62E-01**
80-400	5,58E-01	5,73E-01	5,69E-01	5,16E-01	5,10E-01	**6,25E-01**
80-600	4,82E-01	5,20E-01	5,26E-01	4,73E-01	4,90E-01	**5,99E-01**
80-800	5,10E-01	5,51E-01	5,55E-01	4,95E-01	5,21E-01	**5,66E-01**
80-1000	6,16E-01	6,67E-01	6,56E-01	6,35E-01	6,44E-01	**7,35E-01**
100-200	5,01E-01	5,10E-01	4,98E-01	4,82E-01	4,62E-01	**5,83E-01**
100-400	4,35E-01	4,63E-01	4,52E-01	4,08E-01	4,26E-01	**4,94E-01**
100-600	5,41E-01	5,84E-01	5,87E-01	5,46E-01	5,62E-01	**6,47E-01**
100-800	7,38E-01	7,54E-01	7,48E-01	6,60E-01	7,16E-01	**7,88E-01**
100-1000	4,66E-01	4,91E-01	4,92E-01	4,38E-01	4,53E-01	**5,27E-01**
The best QoS fitness values are in bold						

The convergence curves of the different algorithms vs the number of function evaluations for QoS are shown in Figure 9 with the number of tasks equal to 100 and the number of micro services equal to 1000. The number of iterations is assumed to be the same as the number of tasks. The number of function evaluations in each iteration is equal to the number of individuals in a population, which is 30. Our experiments show that all algorithms suffer from stagnation, except the proposed algorithm DDPASO which succeeds to escape from stagnation.

Figure 10 shows the Computation time comparison in seconds of DDAPSO compared to the other five algorithms for a number of tasks n=20 and a number of concrete services m which varies from 200 to 1000 (the number of iterations is fixed to 100). As seen from Figure 5, the execution time of DDAPSO with respect to 1000 concrete services was 2,1 s, whereas the execution time of MVO, GA, DDA, PSO and ABC were 0,7558794, 0,594872, 3,0191281, 1,7630176 and 3,5550047 s, respectively. From Figure 10, we can observe that DDAPSO is in the DDAPSO is in the middle and presents average time of computation in comparison with the other algorithms.

Table 4 shows that DDAPSO obtains the best results for 25 out of 25 datasets, i.e. DDAPSO performed best on all datasets while MVO, GA, DDA, PSO, and ABC failed to provide any best solution.

6. CONCLUSION AND FUTURE WORK

Users can access IoT computing services via the internet. However, different services with comparable functionality but varying levels of service quality exist. As a result, selecting the suitable services among the service sets to meet the user's needs is a difficult problem. In this paper we have proposed a hybrid algorithm which combines discrete dragonfly algorithm and particle swarm optimization for service selection and composition in the IoT context.

Both the exploration and exploitation stages of this algorithm are improved. Simulation results show that the proposed approach outperforms competing algorithms, especially when dealing with high-dimensional and service composition problems with SLA constraints.

In future research, the service selection and composition problem for the Internet of Things will be seen as a multi-objective optimization problem, with the goal of developing a novel multi-objective optimization method to improve the service selection and composition with QoS.

REFERENCES

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials, 17(4): 2347-2376. http://dx.doi.org/10.1109/COMST.2015.2444095

[2] Giusto, D., Iera, A., Morabito, G., Atzori, L. (2010). The Internet of Things: 20th Tyrrhenian workshop on digital communications. Springer Science & Business Media. http://dx.doi.org/10.1007/978-1-4419-1674-7

[3] Jatoth, C., Gangadharan, G.R., Buyya, R. (2015). Computational intelligence based QoS-aware web service composition: a systematic literature review. IEEE Transactions on Services Computing, 10(3): 475-492. http://dx.doi.org/10.1109/TSC.2015.2473840

[4] Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4): 1053-1073.

https://doi.org/10.1007/s00521-015-1920-1
[5] Kennedy, J., Eberhart, R. (1995). Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942-1948. http://dx.doi.org/10.1109/ICNN.1995.488968
[6] Jatoth, C., Gangadharan, G.R., Buyya, R. (2015). Computational intelligence based QoS-aware web service composition: A systematic literature review. IEEE Transactions on Services Computing, 10(3): 475-492. http://dx.doi.org/10.1109/TSC.2015.2473840
[7] Asghari, P., Rahmani, A.M., Javadi, H.H.S. (2018). Service composition approaches in IoT: A systematic review. Journal of Network and Computer Applications, 120: 61-77. https://doi.org/10.1016/j.jnca.2018.07.013
[8] Zhang, X., Geng, J., Ma, J., Liu, H., Niu, X., Mao, W. (2021). A hybrid service selection optimization algorithm in Internet of Things. EURASIP Journal on Wireless Communications and Networking, 4. http://dx.doi.org/10.1186/s13638-020-01883-2
[9] Hosseinzadeh, M., Tho, Q.T., Ali, S., Rahmani, A.M., Souri, A., Norouzi, M., Huynh, B. (2020). A hybrid service selection and composition model for cloud-edge computing in the Internet of Things. IEEE Access, 8: 85939-85949. http://dx.doi.org/10.1109/ACCESS.2020.2992262
[10] Alizadeh, P., Osmani, A., Khanouche, M.E., Chibani, A., Amirat, Y. (2020). Reinforcement learning for interactive QoS-aware services composition. IEEE Systems Journal, 15(1): 1098-1108. http://dx.doi.org/10.1109/JSYST.2020.2997069
[11] Khanouche, M.E., Atmani, N., Cherifi, A. (2020). Improved teaching learning-based QoS-aware services composition for Internet of Things. IEEE Systems Journal, 14(3): 4155-4164. http://dx.doi.org/10.1109/JSYST.2019.2960677
[12] Khanouche, M.E., Amirat, Y., Chibani, A., Kerkar, M., Yachir, A. (2016). Energy-centered and QoS-aware services selection for Internet of Things. IEEE Transactions on Automation Science and Engineering, 13(3): 1256-1269. http://dx.doi.org/10.1109/TASE.2016.2539240
[13] Kurdi, H., Ezzat, F., Altoaimy, L., Ahmed, S.H., Youcef-Toumi, K. (2018). MultiCuckoo: multi-cloud service composition using a cuckoo-inspired algorithm for the Internet of Things Applications. IEEE Access, 6: 56737-56749. http://dx.doi.org/10.1109/ACCESS.2018.2872744
[14] Alsaryrah, O., Mashal, I., Chung, T.Y. (2018). Bi-objective optimization for energy aware Internet of Things service composition. IEEE Access, 6: 26809-26819. http://dx.doi.org/10.1109/ACCESS.2018.2836334
[15] Li, Q., Dou, R., Chen, F., Nan, G. (2014). A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of Things. International Journal of Computational Intelligence Systems, 7(sup2): 26-34. https://doi.org/10.1080/18756891.2014.947090
[16] Kashyap, N., Kumari, A.C., Chhikara, R. (2020). Multi-objective optimization using NSGA II for service composition in IoT. Procedia Computer Science, 167 1928-1933. https://doi.org/10.1016/j.procs.2020.03.214
[17] Purohit, L., Kumar, S. (2019). Web services in the Internet of Things and smart cities: A case study on classification techniques. IEEE Consumer Electronics Magazine, 8(2): 39-43. https://doi.org/10.1109/MCE.2018.2880808
[18] Zhou, J., Gao, L., Yao, X., Zhang, C., Chan, F.T., Lin, Y. (2019). Evolutionary algorithms for many-objective cloud service composition: Performance assessments and comparisons. Swarm and Evolutionary Computation, 51: 100605. https://doi.org/10.1016/j.swevo.2019.100605
[19] Mirjalili, S., Mirjalili, S.M., Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. Neural Computing and Applications, 27(2): 495-513. http://dx.doi.org/10.1007/s00521-015-1870-7
[20] Yaghoubi, M., Maroosi, A. (2020). Simulation and modeling of an improved multi-verse optimization algorithm for QoS-aware web service composition with service level agreements in the cloud environments. Simulation Modelling Practice and Theory, 103: 102090. https://doi.org/10.1016/j.simpat.2020.102090
[21] Golberg, D.E. (1989). Genetic algorithms in search, optimization, and machine learning. Addion Wesley, 1989(102): 36. https://doi.org/10.1023/A:1022602019183
[22] Wang, D., Yang, Y., Mi, Z. (2015). A genetic-based approach to web service composition in geo-distributed cloud environment. Computers & Electrical Engineering, 43: 129-141. https://doi.org/10.1016/j.compeleceng.2014.10.008
[23] Karaboga, D., Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In International Fuzzy Systems Association World Congress, pp. 789-798. https://doi.org/10.1007/978-3-540-72950-1_77
[24] Al-Masri, E., Mahmoud, Q.H. (2007). Qos-based discovery and ranking of web services. In 2007 16th International Conference on Computer Communications and Networks, pp. 529-534. http://dx.doi.org/10.1109/ICCCN.2007.4317873