People's Democratic Republic Of Algeria
Ministry Of Higher Education and Scientific Research

University of Abderrahmane MIRA of Bejaia
Exact Sciences Faculty
Computer Science department

MASTER RESEARCH THESIS

In

Computer Science

Option

Artificial Intelligence

Theme

Services composition algorithm based on Neural Network in smarts environments

---

Presented by:

Miss. Asma BATROUNI

Miss. Dounia CHERFA

Evaluated on October 1st, 2020. Examining Committee composed of :

Examiner	Dr Houda EL BOUHISSI	MCA	U. A/Mira Bejaia.
Examiner	Dr Achour ACHROUFENE	MCB	U. A/Mira Bejaia.
Supervisor	Dr Mohamed Essaid KHANOUCHE	MCA	U. A/Mira Bejaia.

2019 - 2020

Acknowledgements

Above all, praise is to Allah for the many blessings that we know, and also to his blessings that are absent from our minds.

We wish to express our gratitude to our supervisor *Mr. M. E. KHANOUCHE* for his insightful discussions and availability.

We would also like to thank *Ms. A. CHERIFI* for her moral support and for providing us with constructive suggestions and valuable input.

We thank *Mrs. H. EL BOUHISSI* and *Mr. A. ACHROUFENE* for the interest they have shown in this thesis by agreeing to be part of the jury as examiners.

We would also like to express full appreciation to the people who we have been fortunate enough to get the motivation and guidance from, in the university of Bejaia.

Carrying out this thesis without the spirit of master students of AI would have been more difficult, we thank you for making studies more enjoyable and for letting us escape from the negative atmosphere from time to time.

Last in order but not of importance, overwhelmed with humility and gratefulness, we are thankful to every person who helped in the making of this thesis directly or indirectly.

Dedications

I dedicate this modest work as a sign of reverence, appreciation and gratitude to my dear parents, who painstakingly laid the foundation for my education giving it all it takes, without you I could not achieve all that I have.

My utmost thanks go to my twin sister Sara, who is always by my side encouraging and helping me. Thank you for being the best sister whom I could ever have.

My greatest gratitude goes to the source of my strength, my brothers who are always next to me without you I couldn't be the one who I am now.

My sincere thanks to my grandparents, the BATROUNI and the DJENNADI family.

With great happiness and joy, I share this modest work with my beloved uncles, my close friends Hanane KHELOUF, Yasmine IDIR, Nouara GUEBGUID, Fatima AITSLIANE schoolmates, Childhood friends and to all the ones whom I know in my university journey.

Special thanks to my partner Dounia and the CHERFA family.

I extend my sincere thanks to all the ones who helped me and supported me during my journey, may God bless you.

BATROUNI Asma

Dedications

I dedicate this thesis to my brother, whose memory will last like stars and whose precious smile and warm heart will not be forgotten.

First and foremost, to the pillars of my life, to my parents who have always been there for me, to whom I owe a great debt forever. No amount of words or actions is enough to pay you back for the sacrifices you made and the dreams you abandoned for us.

Next, to my brothers, sister and brother-in-law, I always feel blessed to have you in my life, thank you for being the wonderful people you are and for supporting me. Without forgetting the one and only Aisha Aya, welcome to the family dear cherry.

I also want to thank eternally MY FAMILY for their constant love and unlimited support, as well as extend special thanks to the "Mathelemes 2014-2016", "MoonLight squad" and my loyal friends to whom I wish tremendous success in the future.

To my partner Asma and the whole BATROUNI family.

Last but not least, I would like to express my warmest thanks to all those who were part of my journey, from whom I got the motivation and advice and who accompanied me during my graduate studies, may Allah bless you all.

CHERFA Dounia

Contents

List of Figures

List of Tables

List of Algorithms

Notations and symbols

ABC	**A**rtificial **B**ee **C**olony
AI	**A**rtificial **I**ntelligence
ANNs	**A**rtificial **N**eural **N**etworks
COG	**C**enter **O**f **G**ravity
DE	**D**ifferential **E**volution
DL	**D**eep **L**earning
DRL	**D**eep **R**einforcement **L**earning
FOA	**F**ruit fly **O**ptimization **A**lgorithm
GA	**G**enetic **A**lgorithm
GABC	**D**iscrete **G**best-guided **A**rtificial **B**ee **C**olony
GS	**G**ale **S**hapley
HS	**H**armony **S**earch
IABC	**I**mproved **A**rtificial **B**ee **C**olony
ITL-QCA	**I**mproved **T**eaching **L**earning-based **Q**oS-aware services **C**omposition **A**lgorithm
LSTMs	**L**ong **S**hort **T**erm **M**emories
mABC	**m**odified **A**rtificial **B**ee **C**olony
mABC-QCA	**m**odified **A**rtificial **B**ee **C**olony **Q**oS-aware services **C**omposition **A**lgorithm
MGWO	**M**odified **G**rey **W**olf **O**ptimizer
NNA	**N**eural **N**etwork **A**lgorithm
OS	**O**perating **S**ystem
QCA	**Q**oS-aware services **C**omposition **A**lgorithm
QoS	**Q**uality of **S**ervice
QP	**Q**oS **P**redictor
RL	**R**einforcement **L**earning
SC2N	**S**ervices **C**omposition based on **N**eural **N**etwork
SCOS	**S**ervices **C**omposition and **O**ptimal **S**election
SDG	**S**ervice **D**ependency **G**raph
SGO	**S**ocial **G**roup **O**ptimization
SG-QCA	**S**ocial **G**roup **O**ptimization-based **Q**oS-aware agents services **C**omposition **A**lgorithm
TBA	**T**hreshold **B**ased **A**lgorithm
TF	**T**ransfer **F**unction
WSDL	**W**eb **S**ervices **D**escription **L**anguage

Chapter 1

Introduction

Contents

1.1 Context and motivations

Over the past decade, digital technologies have undergone an unusual growth leading to the emergence of what is called a smart environment. The latter is an ecosystem where users constantly interact with smart objects (sensors, smartphones, appliances, etc.) to improve their everyday life. The intelligence of a smart environment lies in its ability to perceive and use knowledge about an environment as well as adapting to its users to improve their experience in a non-intrusive way. This type of smartness requires the cooperation of several smart devices offering different services to satisfy the increasing user's requirements [Marquardt and Uhrmacher, 2008, Nazerfard and Cook, 2012, Guelzim et al., 2016]. Services are software entities that provide one or more functionalities defined by an interface that includes functional and non-functional properties of the service and allows users to identify the services that best matches their requirements. The services providing basic functionalities and whose implementation is autonomous and does not require the invocation of any other service are *atomic services* and the services whose execution requires the invocation of other services are *composite services*. Since the user's requirements often exceed a single service, the need for a composition is in high demand; the composition process consists of building new services by combining existing services (atomic or composite) to meet the user's demand [Zeng et al., 2003].

Carrying out a manual composition in a large services space is a complex process. Accordingly, the automatic composition is required to allow an automatic services composition for given requests. In addition, with the rapid growth in the number of services that are characterized by the same functionalities, Quality of Service (QoS) has become an important factor in distinguishing between functionally equivalent services. However, the challenge of selecting the best set of services to compose taking into account the global QoS constraints

imposed by the user has become an NP-hard problem [Mabrouk et al., 2009] for which various approaches have been proposed in the literature. These algorithms can be classified into: (i) QoS-aware services composition approaches and (ii) hybrid services composition approaches [Bekkouche, 2018]. The QoS-aware services composition approaches are able to find optimal or quasi-optimal compositions considering the user's constraints in terms of QoS. This category of approaches assumes that the workflow between *abstract services* already exists, where each abstract service groups all concrete services that are functionally equivalent but different in their QoS attributes' values. In the hybrid services composition approaches, in addition to dealing with QoS constraints, they also build the workflow between the abstract services. Furthermore, the hybrid services composition approaches studied can also be grouped into two categories that are graph-based approaches and Artificial Intelligence (AI) approaches.

In this thesis, we address the problem of QoS-aware services composition in smart environments where we propose a services composition approach that falls under the category of the AI approaches; an approach that is not only Qos-aware like all the studied approaches but, unlike some of these approaches scalable and requires reasonable time when finding an optimal or quasi-optimal composition in terms of non-functional requirements, while the functional user's requirements are satisfied.

1.2 Contribution

In this thesis, the Services Composition algorithm based on Neural Network (SC2N) is proposed to find composite services that optimize the value of the overall QoS while satisfying the constraints specified by the user within a reasonable time. The Neural Network Algorithm (NNA) [Sadollah et al., 2018] is able to find quality solutions when encountering an NP-hard optimization problem, has a mature convergence and does not require efforts for fine tuning the initial parameters thus preventing performance issues. Indeed, the fewer parameters to initialize, the more control over the results. Like other meta-heuristics, the SC2N algorithm starts with a randomly generated initial population of compositions and the goal is to move the compositions to the best composition by modifying the value of the weights and the indexes of the candidate services.

1.3 Outline

This thesis is structured as follow:

Chapter 1 is intended to help the understanding of the context, problematic and motivations of the thesis followed by the outline of this thesis.

Chapter 2 gives a state of the art regarding automatic services composition approaches which are classified into two main categories according to the existence of the composition workflow. Then, a comparison of the presented approaches will be given.

Chapter 3 presents the SC2N algorithm that is proposed to solve the problem of QoS-aware automatic services composition.

Chapter 4 is devoted to the evaluation of the SC2N algorithm through several simulation scenarios and to its comparison with an other approach proposed in the literature.

Finally, we conclude this thesis with a conclusion and perspectives.

Chapter 2

State of the art

Contents

2.1 Introduction

This chapter is organized as follows. First, an overview of automated services composition approaches will be presented. Then, the evaluation criteria used in the comparison are identified. Finally, a comparison of the presented approaches will be undertaken according to how they meet the evaluation criteria.

2.2 State of the art

A multitude of approaches have been proposed in the literature to solve the problem of QoS-aware services composition. The studied approaches were grouped into two distinct categories: QoS-aware services composition approaches and hybrid services composition approaches [Bekkouche, 2018].

QoS-aware services composition approaches: These approaches take into account not only the functional characteristics of the services, but also the non-functional ones. The QoS

aspect in services composition is a major factor in the discovery and selection of services since QoS-aware approaches do not only answer to the requests of the user but also guarantee the best possible quality [Baryannis and Plexousakis, 2010].

Hybrid services composition approaches: Unlike the QoS-aware services composition approaches that assume the prior existence of the workflow between abstract services, the hybrid approaches deal with both the QoS aspect and the generation of the services composition workflow.

2.2.1 QoS-aware services composition approaches

An extended gale shapley approach

In [Li et al., 2018b], an approach based on the extended algorithm of Gale Shapley (GS) is proposed to solve the services composition problem. This approach has three stages:

1) Initialization: Every sub-task can be completed by a set of candidate services. Li et al. assumed that the first set of candidate services represents a provider set of services, the last set serves as a users' set. Then, the other sets of candidate services are both providers and users.

2) Preference Ordering: It should be noted that the QoS attributes taken into account in this approach are: time (T), cost (C), reliability (R), cooperation intensity (Co) that represents the cooperation time between two enterprises, energy consumption (E), and that the QoS attributes are divided into two parts: (i) Independent indices that include $\{T, C, R, E\}$ and (ii) a non-independent index Co. In this phase, the preference ordering of each service towards other services belonging to the adjacent set is based on the performance of the independent indices and the non-independent index. Then, relying on the preference list, the GS algorithm is employed between each pair of adjacent candidate sets.

3) Establishment of services composition and optimal selection schemes: Through the above processes of service matching among several different bilateral models, the combination of the services from the first set to the last set is established. Hence, multiple services composition schemes with different QoS values can be obtained. At the end, the best or several better composition schemes are chosen as the services composition solutions. The existence of multiple schemes provide alternatives which can increase robustness and can also be scheduled to different tasks with the same requirements.

Discussion: Athough the performances of the extended (GS) algorithm-based approach show good scalability, low cost, increased reliability, and low energy consumption, however, it also shows low availability, high response time.

QoS-aware services composition approach using evolutionary algorithms

In [Seghir and Khababa, 2018], a hybrid approach using Genetic Algorithm (GA) and Fruit fly Optimization Algorithm (FOA), is proposed to solve the QoS-aware services composition. The

following are the detailed steps of this approach:

1) Initialization phase: Using a heuristic local selection method, an improved initial population was generated to speed-up the convergence of their proposed algorithm.

2) The stop criterion phase: If the composite service with the highest fitness value does not improve over the already fixed number of generations max_gen_imp, the algorithm stops and returns the near-optimal solution, otherwise, the genetic phase will proceed.

3) The genetic phase: In this phase, a novel roulette wheel selection operator is performed; it is based on the selection's probability related to the fitness of the services and its diversities at the same time, where fitted and diversified composite services are typically more likely to be selected for reproduction. After selecting two composite services (parents) in the current population using the above selection operator, a crossover operator produces two new composite services (children) by combining the candidate services of the two selected parents. Afterwards, to maintain the diversity of the population, the mutation operator is used to select a service in the individual obtained by the crossover and randomly replaces it with a new chosen concrete service in the corresponding candidate set of services.

4) FOA phase: This phase is incorporated as a local search strategy; for each individual *indi* generated after the genetic phase, the following steps will be performed:

1. Calculate the selecting probability P_s of *indi*.
2. Generate a random number $r \in [0, 1]$.
3. If r is smaller than P_s, SN neighbours are generated around *indi* to construct a sub-population SubP_i. Then *indi* is replaced with the best individual $Best\_Ind$ in the corresponding sub-population $SubP_i$ (i.e. if $Best\_Ind$ has bigger fitness than *indi*); otherwise, *indi* remains unchanged.

5) The elitism phase: To replace the old population Pop with the new population Pop_{new}, the elitism strategy is used as a strategy of preserving and replacing to speed up the convergence of the algorithm and prevent the loss of the best individuals during the evolutionary process.

6) Verification of the stopping criteria: In this phase, the stopping conditions will be once again checked.

Discussion: This approach has significant performances in comparison with the other algorithms. However, inter-dependencies and correlations among cloud services and also the influence of the distribution of cloud services on distributed clouds were not considered.

Services composition and optimal selection using fuzzy logic approach

In [Li et al., 2018a], the trustworthy method and fuzzy soft set-based decision-making method were aggregated and enhanced with volatility analysis. This approach has three main steps detailed as follow:

1) Trust filtration phase: In this phase, services are filtered based on trust evaluation:

(i) *Direct trust:* when a user receives a series of responses then selects a set of services from candidates.

(ii) *Recommended trust:* when a user selects a set of services from candidates with the help of a partner.

2) Prospect evaluation phase: It introduces the decision-making method used in fuzzy soft set-based decision making. First, a trustworthy of services for Services Composition and Optimal Selection (SCOS) is formed using the users' and their partners' respond (trust), then a correlation-aware services composition strategy is executed to form combinations between the services. Afterwards, the sets of combinations are completed with random composition. On this basis, prospects of combinations are formed based on the integration strategy. After their standardization, fuzzy soft sets operate the decision-making according to level soft set based strategy.

3) Volatility analysis phase: This phase evaluates services by calculating for each one its execution time, quality, cost, and support process so that services composition and optimal selection scale gets reduced. Once the user gets the result of his request, he sends feedback of its request whether it is trustworthy or not.

Discussion: Li et al. improved the decision-making method based on fuzzy soft by exploiting the interaction with the users and although this approach shows good scalability and low cost however, it provides low reliability and low availability.

QoS-aware services composition approach using a population-based algorithm

In [Khanouche et al., 2019b], the Social Group Optimization (SGO) method is opted to resolve the services composition in the context of large-scale environments, as it requires few initialization parameters resulting in a Social Group optimization-based QoS-aware agents services Composition Algorithm (SG-QCA). The approach is implemented in three phases: Initialization phase, improving phase and acquiring phase.

1) Initialization phase: At this stage, the parameters of the SG-QCA algorithm are initialized: the randomly created initial population of compositions, the maximum number of iterations and the number of compositions in the population.

2) Improving phase: A utility value in terms of QoS was used to assess compositions and as enhancing compositions in terms of utility value and the composition itself might not lead to satisfactory performance as some compositions may have a low utility value, Khanouche et al. have opted for the mean of the compositions in order to assess and improve them in an interval with high utility values. At last, when the new utility value of a composition is better than its old one, the composition will be replaced in the population.

3) Acquiring phase: Each composition is enhanced according to a randomly chosen composition; if the new utility value of the composition is better than its old one, it will be replaced in the population.

Discussion: In their simulation results for both randomly generated and real datasets, near-to-optimal compositions were found while reducing the composition time. However, simulation isn't the best option for validating the performances, and it would have been better to compare it with more than one approach.

An enhanced artificial bee colony algorithm for an optimal selection

In [Dahan et al., 2019], an additional enhancement to the Artificial Bee Colony (ABC) approach was proposed for the QoS-aware services selection problem to balance the exploration and exploitation mechanisms of the ABC algorithm.

1) Selection and replacement process: This phase selects a random composition from the generated compositions of the ABC algorithm, then a random service of the latter is replaced by a neighboring service. In the early iterations, the algorithm selects a service randomly from the farthest neighbors for replacement, then it selects a service randomly from the nearest neighbors. Afterward, euclidean distance is used to calculate the distance to the randomly selected SN (Number of Services) neighbors. It should be noted that the distances of the selected services and the best service are stored.

2) Swapping process: This process improves the population by swapping between the services of the two best solutions of the previous phase to generate two new solutions and only the two better solutions in terms of the utility value are retained.

Discussion: The proposed method successfully enhanced the ABC algorithm performance and showed significant performances in comparison with the ABC approach.

QoS-aware services composition using an improved teaching learning-based optimization approach

In [Khanouche et al., 2019a], an Improved Teaching Learning-based QoS-aware services Composition Algorithm (ITL-QCA). This approach consists of four phases:

1) Initialization phase: The initial population of compositions, the number of iterations $Maxiter$, the number of compositions in the population $Popsize$, and the learning probability P_c are initialized. As for the population matrix, it is rearranged after a given number of iterations P_m iterations.

2) Teaching Phase: In this phase the utility value of each composition in the population is evaluated to find the best composite service that has the highest utility value.

3) Learning phase: In this phase, a random number r is generated for each composition in the population. If the generated number r is less than P_c , each composition CC_l is improved

according to the best neighbor composition CC_n in terms of utility value and a randomly neighbor composition CC_r. If r is greater than P_c, the composition CC_l is improved according to a randomly selected composition CC_e in the population.

4) Self-learning phase: In this phase, each composition CC_l is improved according to its historical information in the present and the previous iterations. Indeed, if the composition CC_i has been changed over the two consecutive iterations, it gets improved according to its components of the iterations but if the composition CC_l has not been changed over the two iterations, the composition is improved using the *normrnd* function that returns a random number from the normal distribution with a mean and a standard deviation and takes the average of the best composition services, the mean composition, the absolute value of the difference between the best composition and the mean composition as parameters.

Discussion: The proposed approach shows good performances in terms of utility values and composition time, however it does not handle the mobility in the composition context while considering different composition structures.

QoS-aware for services composition using a modified artificial bee colony algorithm

In [Chandra and Niyogi, 2019], a Modified Artificial Bee Colony (mABC) algorithm was proposed for the QoS-aware service selection problem. The approach is carried as follows:

1) Initialization phase: In this phase, a chaotic-based opposition learning methodology is used to generate a better initial population and improve the exploration mechanisms of the ABC algorithm.

2) Search equation for employed bees: A new search equation has been introduced in order to make an exhaustive search of the generated compositions in order to select the best composition.

3) Search equation for onlooker bees: After receiving the selected services chosen in the previous phase, Chandra et al. improve them by introducing a differential evolution strategy equation. This phase calculates and compares the fitness values of the best-found compositions in order to retain only the best services composition.

Discussion: The proposed approach shows good performances in terms of composition time and scalability. However, Chandra et al. did not take into account to the problem that the discrete features of the quality of service compositions.

2.2.2 Hybrid services composition approaches

Some of the Hybrid services composition approaches we have studied can also be grouped into two categories, that are graph-based approaches and Artificial Intelligence (AI) approaches.

2.2.2.1 Graph-based approaches

Graph-based approaches construct a graph by considering the semantic input/output relationships of services, then apply graph search techniques for the selection of services, which is considered the shortest path problem under constraints of global QoS [Bekkouche et al., 2017].

Hybrid optimization algorithm for large-Scale QoS-aware services composition

In [Rodriguez-Mier et al., 2015], a hybrid approach for automatic services composition was proposed. It generates semantic input-output based compositions with optimal end-to-end QoS while minimizing the number of services of the resulting composition. It has four main steps:

1) Generation of the Service Match Graph: After receiving the inputs/outputs provided by the user, all relevant services that can be part of the final composition will be identified in addition to all possible matches between their inputs and outputs. Thus, creating a graph to match the service.

2) Optimal end-to-end QoS: Once the Service Match Graph is computed for a composition request, next, a generalized Dijkstra based label-setting algorithm was used for computation of the optimal composition that minimizes a single objective QoS function.

3) Graph optimizations: To do so, Rodriguez-Mier et al. applied a set of optimizations to reduce the search space. The different phases that are sequentially applied are:
(i) Elimination of services from the graph, that do not contribute to the outputs of the request.
(ii) Pruning of services that cannot be part of any optimal QoS composition.
(iii) The third and the fourth pass analyze service equivalences and dominances in the Service Match Graph in order to combine those that are equivalent, or replace those that are dominated.

4) Hybrid local-global search: This step aims at extracting the optimal QoS with the minimum number of services. After the local search is performed to extract a good solution, the global search is performed in the remaining time to obtain a better solution by exhaustively exploring the space of possible solutions.

Discussion: As global research seeks to obtain a composition with a minimum of services, it is not temporal, likewise for the construction of the graph at runtime.

QoS-aware optimal and automated semantic approach

In [Bekkouche et al., 2017], an approach based on a planning graph and a Harmony Search (HS) algorithm to select the optimal or quasi-optimal solution in semantic services composition is proposed.

For a user request given in terms of functional and non-functional requirements. First of all, according to functional needs, a planning graph is constructed to code all the possible

composition solutions by calculating the semantic similarity between the output parameters and the input parameters of certain services in different layers. After that, the HS algorithm uses the functional properties and the QoS to find the optimal composition solution. It should be noted that when matching the HS algorithm with the composition problem of the Web service: a harmony represents a candidate composition solution, a harmony memory represents a population of compositions, a musical pitch represents an abstract service and an aesthetic standard represents a functional physical form that evaluates each composition.

A set of compositions is randomly generated to be the harmony memory, then the compositions are updated like this: If a random number generated is less than a given HMCR probability, each candidate service of the composition is replaced by a selected service candidate of the current generation, so that the utility function is maximized. Otherwise, the candidate service will be replaced by a candidate service selected at random from the same abstract service, then evaluated using the fitness function.

This process is repeated until the defined termination criterion is met and the optimal or near-optimal composition is returned.

Discussion: Other control structures such as conditional choices, loops, etc. were not taken into account in this approach, in addition, better results could be obtained if the pre/post conditions were considered for a better expression of the functionalities of the services.

Automated composition based on abstract services

In [Fki et al., 2017], an automated and flexible composition based on abstract services for a better adaptation to user intentions is proposed. To benefit from the user requirements, the notion of intention has been introduced, with reference to the combination of a goal and a set of constraints (functional and non-functional ones) expressing the way that this goal is achieved. This approach is carried out in four consecutive steps:

1) Intentions graph enrichment: This step takes as input user requirements materialized in a graph of intentions that is assumingly complete and includes all necessary information for enrichment. Then, this graph is enriched by applying two rules, to identify the relationship between intentions that do not dominate other intentions.

Rule R_1: identifies an intention x that precedes an intention y and intention z dominated by x, then adds a precedence relation between z and y.

Rule R_2: identifies an intention x that precedes an intention y and the intention y dominates the intention z , then adds a precedence relation between x and z.

2) Generation of initial composition schema: The enriched graph is used to generate an initial composition schema by applying these steps ahead:

(i) Building control flow: It is inferred automatically from dependencies expressed in the intention's specification.

(ii) Selection of abstract services: A score of matching between the intention and the abstract service is calculated to find a set of appropriate candidate abstract services that meet an intention. This matching is based on the use of ontologies and the degree of semantic similarity. Then, the abstract services having a matching score that exceeds a threshold are added

to the set of candidate abstract services. After finding candidate abstract service sets, all different alternatives for the initial composition schema are extracted. Finally, the selection of the adequate alternative abstract services is based on the semantic connection quality (relation between input and output parameters) between services.

3) Generation of final composition schema: Using a refinement mechanism of abstract services based on semantic matching taking into account user context and constraints, the initial composition schema will be further refined: more abstract services of finer granularity will be selected iteratively until obtaining a final composition schema formed by atomic abstract services.

4) Generation of the execution plan: The concrete services that can fulfill the obtained composition schema are selected according to required non functional constraints and assigned to each involving task. Thus, the execution plan is generated.

Discussion: Their approach can specify generic processes for different situations, and select adequate services that meet user intentions. However, it defines only the set of activities and the control flow among them, and not the concrete services to be invoked. It also assumes that the intention specifications are available and did not extract them.

Dynamic services composition using AND/OR directed graph

In [Elmaghraoui et al., 2017], focused on reducing the complexity and time needed to create and execute a semantic services composition; the proposed approach reused some components from an already existing platform in precise the discovery engine with the motive to enhance the composition module by taking into consideration the non-functional requirements. The approach performs as follow:

At the design time: Using the information generated by the matchmaking and discovery engines, the Service Dependency Graph (SDG) generator computes the graph with all the semantic relations (Exact, Plugin) between the inputs/Outputs of all the available services known to the registries used. However, to speed up the calculation of the graph, beforehand creating it; a pre-computing and indexing of relevant services are applied to the group to reduce the number of services and relations.

Then, the created graph has to be optimized. To do this, Elmaghraoui et al. analysed the dominance between the services to seek out those that are equivalent or higher than others and replaced the initial services of the graph by the abstract services that capture the functionality of the dominant or equivalent services. This reduces the search size of the search space, which means fewer paths to explore during the search. Then, once the optimal composition solution has been generated, the abstract services are replaced by the dominant original services or equivalent by a combination of dominated services that satisfy the functionality of the dominated service. Afterward, a search algorithm is applied over the reduced graph to generate all the shortest paths between every pair of nodes, then stored them for eventual use at the composition runtime.

At runtime: after receiving the user's request (inputs and outputs), Elmaghraoui et al. define the start and end nodes in the graph and extract the precomputed shortest paths between them to get an optimal solution.

Discussion: The use of the concept of abstract services allows flexibility and adaptability without having to create the composition of the service from scratch at the time of execution.

Pre-joined semantic indexing graph

In [Fan et al., 2019], a hybrid approach using fuzzy logic with a graph plan is proposed to optimize services composition. Fan et al. introduced two phases detailed as follow:

1) Fuzzy Control: Firstly, the Fuzzification part converts the exact input values information from the actual system into the fuzzy values for each input and to decompose them into $fuzzysets($ response time, throughput, and global QoS). Then, services are mapped to degrees of membership for linguistic terms using the triangular membership functions. After that, the Decision-Making-Logic part determines how the fuzzy logic operations are performed with the $IF-THEN$ rules based on the Knowledge Base. Where IF part contains various QoS standard member functions of a service, and $THEN$ part contains one of the member functions that reflects the global QoS status of the services. Then, The approach uses the Center Of Gravity (COG) method to defuzzify QoS and obtains an exact output result.

2) Modified GraphPlan: The GraphPlan strategy is modified by adding services prune in the forward phase and selecting optimal candidate services in the backward phase, according to functional requirements and global QoS, wherein the forward phase Fan et al. ignore the less competitive services to the global QoS and in the backward phase, the services are selected according to functional goals and their global QoS.

Discussion: Although their approach has shown a significant performance compared to other algorithms, the plug-in matching degree to match services and solve the composition problem with datasets was not from the real world.

2.2.2.2 AI based approaches

AI-based approaches construct a graph by using artificial intelligence approaches such us clustering the services into groups of clusters to form abstract services, then apply graph search techniques for the selection of services.

A clustering network-based approach to service composition

In [Li et al., 2017], a service clustering network-based services composition approach is proposed. It can be divided into two stages, i.e. offline process and online process.

1. Offline process: Based on their name and functional information, a concept-based similarity calculation method is used to support the service clustering. the services will be clustered

into small-scale abstract services, each of which can be denoted by a node in the network of the abstract services. Once the Abstract services are formed, an abstract network according to the composable relationship between them is built where similarity calculation is performed between the input interface and output interface of different abstract services; two services are perceived as compatible if the degree of interface similarity exceeds a predefined threshold.

2. Online process: For a given task request, an abstract service composition path that matches the execution flow of the task can be obtained according to the breadth-first search algorithm. Then, combining this abstract service composition path with consumer's requirements, a service candidate set for each sub-task can be derived accordingly. After all of the above steps, the service composition solution for a task can be generated by utilizing certain matching and composition algorithms, which has been intensively studied.

Discussion: Searching algorithms are one of the keys for a better services composition approach and the breadth-first is not necessarily the most performing among the searching algorithms.

QoS-aware for services composition using a deep learning approach

In [Labbaci et al., 2017], an approach that relies on deep learning for services composition was proposed and it consists of two phases as follow:

1) Prediction phase: This phase uses the Long Short Term Memories (LSTMs) to predict future QoS and it is decomposed into two prediction categories:

(i) QoS composition requirements: This category specifies the preferred and acceptable intervals for each QoS attribute then compares the result of the fitness function with the predicted QoS values; if the results are under or almost equals to the predicted ones then those services may be acceptable or preferred.

(ii) Predicting Long-Term QoS Trends: This phase uses a trained deep recurrent LSTM with sequences of QoS values observed at different periods of time in order to give predicted intervals of the best composite solutions where QoS predictor (QP) is introduced to manage the QoS prediction for all services and saves the global QoS values in a QoS repository.

2) Long-Term QoS Compliance Checking: Once a request is given, this approach randomly selects a service, then implement it in the prediction phase using two heuristics that are:

(i)Conservative heuristic: calculates for each chosen service S its fitness value that should belong the acceptable or preferred predected intervals.

(ii) The soft compliance heuristic: denotes that the sum of the fitness function of the global composite service should belong to the preferred interval.

Discussion: The proposed approach puts forward a deep learning approach for dynamic QoS based service composition which got promising results compared with existing QoS prediction techniques.

A reinforcement learning approach for services composition

In [Liu et al., 2019],an adaptive service composition based on deep reinforcement learning approach is proposed. Lui et al. decomposed their approach into three phases detailed as follow: This phase aims to maximize the cumulative reward value of action from the environment to learn an action strategy. Lui et al. defined an infinite discount model, where the agents take into account the future infinite steps reward and accumulate it in a value function.

2) Long short-term Memory: Lui et al. adopted an LSTM technique to standard recurrent neurons using mathematical formulas. It simulates the state of services composition problem based on QoS values, each action will update the Q-value table accordingly and store them.

3) Deep reinforcement learning This phase is a combination of DL and RL to form Deep Reinforcement Learning (DRL). It trained the DRL to form a fitted Qvalue surface through the neural network, which is used as an online heuristic function. Lui et al. uses a playback memory unit is used to store samples, and random sampling is used to train network parameters.

Discussion: The objective of the approach is to enable users to obtain high-quality combination schemes to meet user requirements and showed efficient performance but it focuses only on the optimization schemes of composite services.

2.3 Comparative analysis

This section aims to present a comparative study between the automatic services composition approaches presented in this chapter. The evaluation criteria that are used in the comparison are first identified. A summary table is then drawn to compare the approaches according to the identified criteria.

2.3.1 Comparison criteria

Availability of the plan: This criterion refers to the representation of the composition as an abstract workflow where each abstract service can be linked to a concrete service among the set of concrete services that are functionally equivalent [Bekkouche, 2018].

Scalability: Scalability in services composition refers to approaches that maintain a good performance in terms of execution time and QoS of the composition as the number of QoS parameters and abstract/concrete services increases [Anthony Steed, 2009].

QoS-awareness: QoS is a major factor in the service selection since QoS-aware services composition approaches do not only offer the functionalities requested by the user but also guarantee a high quality [Baryannis and Plexousakis, 2010].

Assessment methodology: This criterion refers to the performance evaluation strategy adopted by any services composition approach, whether it is a simulation using synthetic/real-

world datasets or a scenario-based evaluation.

The resolution model: Refers to the techniques used to solve the services composition problem such as graph methods, planification techniques of AI, etc.

2.3.2 Comparison table

Table 2.1 summarizes the approaches presented in this chapter classified according to the following criteria: the resolution model, availability of the plan, QoS-awareness, scalability and assessment methodology.

Table 2.1: Comparative table of automatic services composition approaches.

	Category	Resolution model	Availability of the plan	QoS-awareness	Scalability	Assessment methodology
[Seghir and Khababa, 2018]	QoS-aware services composition	GA and FOA	Yes	Yes	Yes	Simulation
[Li et al., 2018a]		Prediction algorithm	Yes	Yes	Yes	Simulation
[Li et al., 2018b]		GS algorithm	Yes	Yes	Yes	Simulation
[Khanouche et al., 2019b]		SGO algorithm	Yes	Yes	Yes	Simulation
[Dahan et al., 2019]		ABC algorithm	Yes	Yes	Yes	Simulation
[Khanouche et al., 2019a]		ITL algorithm	Yes	Yes	Yes	Simulation
[Chandra and Niyogi, 2019]		mABC algorithm	Yes	Yes	Yes	Web Service Challenge dataset 2007
[Rodriguez-Mier et al., 2015]	Hybrid services composition	Planning graph	No	Yes	No	Web Service Challenge 2009-2010
[Bekkouche et al., 2017]		Planning graph + HS algorithm	No	Yes	Yes	Web Service Challenge dataset 2009
[Fki et al., 2017]		Planning graph	No	Yes	Not addressed	Case study
[Elmaghraoui et al., 2017]		Planning graph	No	Yes but not tested	Not addressed	Simple scenario
[Li et al., 2017]		Clustering network + BF algorithm	No	Yes	Yes	Simulation
[Labbaci et al., 2017]		RNN+LSTM	No	Yes	No	Simulation
[Fan et al., 2019]		Neural network	No	Yes	Yes	Web Service Challenge 2009 dataset
[Liu et al., 2019]		RL+LSTM	No	Yes	Yes	Web Service Challenge 2010 dataset

2.3.3 Discussion

As can be seen in Table 2.1, all of the studied approaches are capable of producing optimal or quasi-optimal solutions satisfying the non-functional constraints (QoS) of the user; in [Seghir and Khababa, 2018], [Li et al., 2018a] and [Dahan et al., 2019], optimal composite services are obtained but they require a high composition time on a large scale environment. In [Li et al., 2018b], [Khanouche et al., 2019b] and [Khanouche et al., 2019a], quasi-optimal composite services are found in a reasonable composition time. Whereas in [Rodriguez-Mier et al., 2015], [Bekkouche et al., 2017], [Fki et al., 2017] and [Elmaghraoui et al., 2017], flexibility and ability to achieve a better performance are shown in their results. Nevertheless, these approaches may not be efficient on a large scale environment because the scalability has not been addressed. Then in [Liu et al., 2019], [Labbaci et al., 2017] and [Li et al., 2017], optimal and quasi-optimal composite services are obtained, but the composition time is not evaluated in these approaches which is an important factor in services composition.

In this research work, we address the QoS-aware services composition challenge using the neural network algorithm [Sadollah et al., 2018]. The NNA algorithm is a meta-heuristic optimization algorithm (population-based) that benefits from the complicated structure of the ANNs and its operators in order to generate new compositions.

Compared to the studied approaches, we propose a QoS-aware services composition approach that finds an optimal or quasi-optimal composition in terms of non-functional requirements within a reasonable time, while the functional user's requirements are satisfied. Furthermore, we propose an approach that is scalable and requires fewer parameters to be adjusted.

2.4 Conclusion

We have presented in this chapter a state of the art on automatic services composition approaches that are classified into two categories: QoS -aware services composition approaches and hybrid services composition approaches. Hybrid approaches are also categorized into two main subcategories that are graph-based approaches and AI-based approaches. In the next chapter, a services composition algorithm based on neural network approach will be proposed and detailed.

Chapter 3

Services Composition algorithm based on Neural Network

Contents

3.1 Introduction

The creation of new value-added services through the process of combining a set of existing services is called services composition. In this chapter, we propose the Services Composition algorithm based on Neural Network (SC2N) to solve the automatic QoS-aware services composition problem by satisfying both user's functional and non-functional requirements.

3.2 The Neural network algorithm

Artificial Neural Networks (ANNs) are computational networks inspired by nervous systems and are used to estimate or approximate functions that can depend on a large number of inputs. ANNs are generally presented as systems of interconnected neurons that are arranged in a series of layers; the input layer receives inputs that then go through one or more hidden layers. The hidden layer's role is to transform the input into something the output layer can use. The learning process in the ANNs generally occurs when weights among layers change when the predicted amounts are significantly different from the calculated amounts. At the end, the trained neural network is capable of predicting outputs that are consistent with new

data collections [Liew, 2016].

Neural network algorithm The NNA algorithm is a meta-heuristic optimization algorithm that exploits the unique structure of the ANN to find new solutions. The NNA algorithm receives input data consisting of a population of pattern solutions (i.e., candidate solutions) and a target data (i.e., best solution), and then tries to map the input data to the target data. Inspired by the ANN, the best solution at each iteration represents the target solution and the goal is to move other pattern solutions to the target solution, knowing that the latter is updated at each iteration [Sadollah et al., 2018]. The process of the NNA algorithm consists of four phases (see Fig. 3.1).

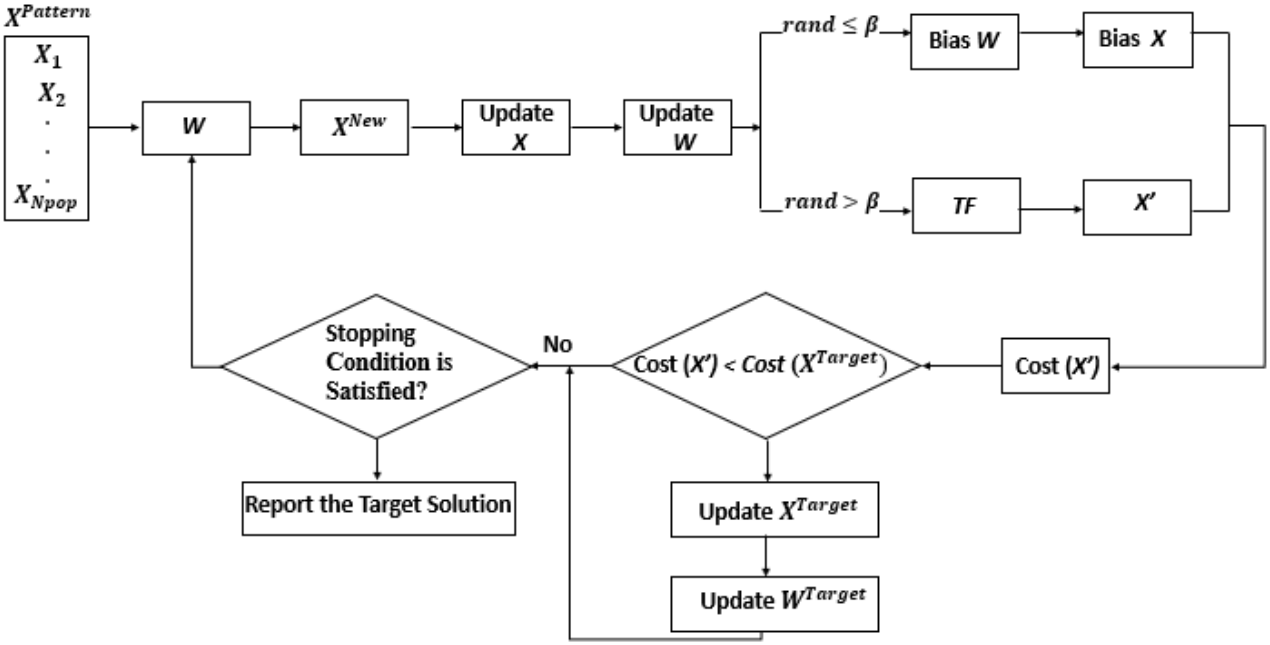

Figure 3.1: Processes of the NNA algorithm.
[Sadollah et al., 2018]

1) Generating the initial population: In a D dimensional optimization problem, each pattern solution is an array of $1 \times D$ defined as follows:

$$Pattern\ Solution = [x_1, \cdots, x_j, \cdots, x_D] \qquad , j = 1, \cdots, D \tag{3.1}$$

In this phase, the parameters of the algorithm are initialized such as the size of the population N_{pop} and the dimension D. Then, an initial population of pattern solutions X is randomly generated and represented as an $N_{pop} \times D$ matrix :

$$X = \begin{bmatrix} x_1^1 & \cdots & x_j^1 & \cdots & x_D^1 \\ \ddots & \vdots & \ddots & \vdots & \\ x_1^i & \cdots & x_j^i & \cdots & x_D^i \\ \ddots & \vdots & \ddots & \vdots & \\ x_1^{N_{pop}} & \cdots & x_j^{N_{pop}} & \cdots & x_D^{N_{pop}} \end{bmatrix}, i = 1, \cdots, N_{pop} \quad , j = 1, \cdots, D \tag{3.2}$$

Each of the decision variable values $(x_1, \cdots, x_j, \cdots, x_D)$ can have real values or can be defined over a set of discrete variables. The cost of a pattern solution is evaluated using a cost function (C) defined as follows:

$$C_i = f(x_1^i, \cdots, x_j^i, \cdots, x_D^i) \qquad , i = 1, \cdots, N_{pop} \quad , j = 1, \cdots, D \tag{3.3}$$

where f is the objective function. Then after calculating the cost function for all pattern solutions, the one with the minimum value (we consider a maximisation problem) of the objective function is considered to be the target solution (X^{Target}).

2) Weight matrix: In the NNA algorithm, initial weights (w_i^k) are defined using the following equation:

$$W(t) = \begin{bmatrix} w_1^1 & \cdots & w_1^k & \cdots & w_1^{N_{pop}} \\ \ddots & \vdots & \ddots & \vdots & \\ w_i^1 & \cdots & w_i^k & \cdots & w_i^{N_{pop}} \\ \ddots & \vdots & \ddots & \vdots & \\ w_{N^{pop}}^1 & \cdots & w_{N_{pop}}^k & \cdots & w_{N_{pop}}^{N_{pop}} \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{k1} & \cdots & w_{N_{pop}1} \\ \ddots & \vdots & \ddots & \vdots & \\ w_{1i} & \cdots & w_{ki} & \cdots & w_{N_{pop}i} \\ \ddots & \vdots & \ddots & \vdots & \\ w_{1N_{pop}} & \cdots & w_{kN_{pop}} & \cdots & w_{N_{pop}N_{pop}} \end{bmatrix}, k, i = 1, \cdots, N_{pop} \tag{3.4}$$

where $W(t)$ is a square matrix $(N_{pop} \times N_{pop})$ of random numbers generated uniformly in the interval $[0, 1]$ during the iteration t.

There is also an imposed constraint for the weight values that is the summation of weights for a pattern solution must be equal to 1 and it can be mathematically defined as follows:

$$\sum_{i=1}^{N_{pop}} w_{ki}(t) = 1 \qquad , k = 1, \cdots, N_{pop} \tag{3.5}$$

As for Eq. (3.6), it states that weight values are belonging to uniformly distributed random numbers in the interval $[0, 1]$.

$$w_{ki}(t) \in \cup(0, 1) \qquad , k, i = 1, \cdots, N_{pop} \tag{3.6}$$

The process seen up until now can be summarized as seen in Algorithm 1.

Algorithm 1: The initial round of the NNA algorithm.

Input: The population size N_{pop},
The size of the dimension D.

Begin
Randomly generate an initial population of pattern solutions;
Calculate the cost of each initial pattern solution;
Find the target solution (X^{Target});
Randomly generate the initial weight matrix;
Find the target weight (W^{Target}) associated to the target solution (X^{Target});
End
Output: The initial population,
(X^{Target}) and (W^{Target}).

When the weight matrix ($W(t)$) is produced, the population of pattern solutions will be updated using the equations:

$$\overrightarrow{X}_i^{New}(t+1) = \sum_{k=1}^{N_{pop}} w_{ki}(t) \times \overrightarrow{X}_k(t) \qquad , i = 1, \cdots, N_{pop} \tag{3.7}$$

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_i(t) + \overrightarrow{X}_i^{New}(t+1) \qquad , i = 1, \cdots, N_{pop} \tag{3.8}$$

Once new pattern solutions are obtained from the previous population, the weight matrix should be updated based on the best weight value called "target weight" while satisfying the constraints (3.5) and (3.6). The following equation updates the weight matrix:

$$\overrightarrow{W}_i^{Updated}(t+1) = \overrightarrow{W}_i(t) + 2 \times rand \times (\overrightarrow{W}^{Target}(t) - \overrightarrow{W}_i(t)) \qquad , i = 1, \cdots, N_{pop} \tag{3.9}$$

The updating of the population and the weight matrix in the NNA algorithm are summarized in Algorithm 2.

Algorithm 2: Updating the population and the weight matrix.

Input: The population of pattern solutions (X), the weight matrix (W),
(X^{Target}) and (W^{Target}).

Begin
Calculate new pattern solutions (X^{new}) using Eq. (3.7);
Update the pattern solutions (X) using Eq. (3.8);
Update the weight matrix (W) using Eq. (3.9);
End
Output: The updated population of patterns,
The updated weight matrix.

3) Bias operator: The role of this operator is preventing the algorithm from premature convergence especially at early iterations by modifying a subset among the updated population of pattern solutions and the updated weight matrix. The Bias operator exploits a modification factor β that determines the percentage of the pattern solutions that should be modified. Its value has been initially set to 1 and will be decreased at each iteration using Eq. (3.10). It should be noted that the reason beta is multiplied times 0.99 is to avoid excessive changes in the pattern solutions until the final iterations.

$$\beta(t+1) = \beta(t) \times 0.99 \qquad , t = 1, \cdots, Max_{Iteration} \tag{3.10}$$

4) Transfer function operator: The objective of this operator is to guarantee the enhancement of the solutions by bringing the pattern solutions closer to the target solution to update and generate better quality solutions using Eq. (3.11) defined as follows:

$$(TF(\overrightarrow{X}_i(t+1))) = \overrightarrow{X}_i(t+1) + 2 \times rand \times (\overrightarrow{X}^{Target}(t) - \overrightarrow{X}_i(t+1)) \qquad , i = 1, \cdots, N_{pop} \tag{3.11}$$

The pseudo code in Algorithm 3 shows the third and fourth phases of the NNA algorithm.

Algorithm 3: The combination of the Bias and TF phases in the NNA algorithm.

Input: The updated population of patterns,
The updated weight matrix.

Begin
for i = *1 to* N_{pop} **do**
  if $rand \leq \beta$ **then**
    ———————- Bias for the new compositions ———————————
    % N_b: No. of biased variables in the population of the new pattern solutions
    % *Round*: a function rounding numbers to a specified level of precision
    $N_b = Round(D \times \beta)$
    % *rand* is a function that generates a sequence of numbers randomly
    % LB and UB represent the lower and upper bounds of a problem
    for j = *1:* N_b **do**
      X^{Input} (i, $Integer\ rand\ [0, D]$) $= LB + (UB - LB) \times rand$
    end
    ———————- Bias for the updated weight matrix———————————
    % N_{wb}: No. of biased variables in the updated weight matrix
    % U is a function that generates numbers randomly between 0 and 1
    $N_{wb} = Round(N_{pop} \times \beta)$
    for $j = 1 : N_{wb}$ **do**
      $W^{Updated}$ (j, $Integer\ rand\ [0, Npop]$) $= U(0, 1)$
    end
  else
    ———————- Transfer Function (TF) operator ———————————-
    Apply the (TF) operator using Eq. (3.11).
  end
end
End
Output: The final population of patterns of current iteration,
(X^{Target}), (W^{Target}).

Finally, the pseudo-code of the NNA algorithm is given below.

Algorithm 4: The NNA algorithm.

Input: The population size (N_{pop}),
The maximum number of iterations ($Max_{iteration}$).

Begin
Perform the initial round of the NNA algorithm using Algorithm 1;
while $Max_{iteration}$ *is not reached* **do**
    The updating of the population and the weight matrix using Algorithm 2;
    The combination of the Bias and TF phases are carried out using Algorithm 3;
    Calculate the cost of all pattern solutions;
    Update the temporal best solution and its corresponding target weight;
    Update the value of β using Eq. (3.10);
End
Output: ($X^{TargetFinal}$),
($W^{TargetFinal}$).

3.3 NNA-based QoS-aware services composition approach

3.3.1 Presentation

The Services Composition algorithm based on Neural Network (SC2N) is proposed, where the services composition problem is modelled and solved using the NNA algorithm. A pattern solution $X_i = (x_1^i, \cdots, x_j^i, \cdots, x_D^i)$ in the NNA algorithm represent a solution composition $CC_i = (cs_1^i, \cdots, cs_j^i, \cdots, cs_D^i)$ in the context of services composition, where cs_j^i is the index of the jth abstract service of the ith composition. It should be emphasized that the best composition CC^{best} corresponds to the composition with the highest aggregated QoS value. Table 3.1 shows the terms matching between the NNA algorithm and QoS-aware services composition problem.

Table 3.1: The terms matching between the NNA algorithm and the QoS-aware services composition problem.

NNA algorithm	**Services composition**
The population of solution patterns	The population of the services compositions
Dimension	The number of abstract services
Pattern solution	Services composition
Cost function	Fitness function in terms of QoS
Target pattern	The best services composition in terms of the fitness value

3.3.2 The SC2N algorithm steps

The steps of the SC2N algorithm work as follows:

1) Initialization step: In this step, the initial population of compositions is randomly generated and represented in the form of a matrix CC of size $(N_{pop} \times numOfAS)$, and it is given as follows:

$$CC = \begin{bmatrix} cs_1^1 & \cdots & cs_j^1 & \cdots & cs_{numOfAS}^1 \\ \ddots & \vdots & \ddots & \vdots & \\ cs_1^i & \cdots & cs_j^i & \cdots & cs_{numOfAS}^i \\ \ddots & \vdots & \ddots & \vdots & \\ cs_1^{Npop} & \cdots & cs_j^{Npop} & \cdots & cs_{numOfAS}^{Npop} \end{bmatrix} i = 1, \cdots, N_{pop} \ , j = 1, \cdots, numOfAS \quad (3.12)$$

When the initial population is obtained, the corresponding aggregated QoS value of each composition is calculated and the best composition CC^{best} with the best QoS value is determined. The weight matrix $(N_{pop} \times N_{pop})$ is also randomly initialized and the target weight W^{Best} corresponding to the best composition CC^{best} will be stored for a future use.

$$W = \begin{bmatrix} w_{11} & \cdots & w_{k1} & \cdots & w_{N_{pop}1} \\ \ddots & \vdots & \ddots & \vdots & \\ w_{1i} & \cdots & w_{ki} & \cdots & w_{N_{pop}i} \\ \ddots & \vdots & \ddots & \vdots & \\ w_{1N_{pop}} & \cdots & w_{kN_{pop}} & \cdots & w_{N_{pop}N_{pop}} \end{bmatrix}, k, i = 1, \cdots, N_{pop} \quad (3.13)$$

The pseudo code in Algorithm 5 shows the first phase of the SC2N algorithm.

Algorithm 5: The initial round of the SC2N algorithm.

Input: The population size N_{pop};
The number of abstract services $numOfAS$.

1 **Begin**
2 Randomly generate an initial population of compositions;
3 Calculate the cost of each initial composition;
4 Find the best solution (CC^{Best});
5 Randomly generate the initial weight matrix;
6 Find the target weight (W^{Best}) associated to the best solution (CC^{Best});
7 **End**
8 **Output:** The initial population,
9 (CC^{Best}) and (W^{Best}).

2) Updating step: In this phase, new compositions are obtained from the previous population using the equations below:

$$\overrightarrow{CC}_i^{New}(t+1) = \sum_{k=1}^{N_{pop}} w_{ki}(t) \times \overrightarrow{CC}_k(t), \qquad i = 1, \cdots, N_{pop} \quad (3.14)$$

$$\overrightarrow{CC}_i(t+1) = \overrightarrow{CC}_i(t) + \overrightarrow{CC}_i^{New}(t+1), \qquad i = 1, \cdots, N_{pop} \quad (3.15)$$

Then, the weight matrix must also be updated based on the target weight using Eq. (3.9) while satisfying the constraints (3.5) and (3.6).

The second phase of the SC2N algorithm is presented in Algorithm 6.

Algorithm 6: Updating the population of compositions and the weight matrix.

Input: The population of compositions,
The weight matrix.

Begin
Generate new compositions (CC^{New}) using Eq. (3.14);
Update the compositions (CC) using Eq. (3.15);
Update the weight matrix (W) using Eq. (3.9);
End
Output: The updated population of compositions,
The updated weight matrix ($W^{Updated}$).

3) Bias operator: This operator allows the diversity of the compositions from one generation to another by exploiting the modification factor β of Eq. (3.10) that determines the percentage of compositions that should be modified.

4) Transfer function operator: This operator is applied to guarantee the enhancement of the compositions by bringing the new compositions closer to the best composition by using the equation below:

$$TF(\overrightarrow{CC}_i(t+1)) = \overrightarrow{CC}_i(t+1) + 2 \times rand \times (\overrightarrow{CC}^{Best}(t) - \overrightarrow{CC}_i(t+1)); \tag{3.16}$$

It is worth mentioning that the aggregated QoS value of all compositions must be recalculated after each phase of the SC2N algorithm and that whenever the candidate services do not respect the bounds of the services composition problem, a $refine$ function is used in order to fix it; if the index of the candidate service is strictly inferior to the lower bound LB (the minimum number of candidate services) then $index(CS) = mod(index(CS), LB)$ and in case the index of the candidate service is strictly superior to the upper bound UB (the maximum number of candidate services) then $index(CS) = mod(index(CS), UB)$.

The pseudo code in Algorithm 7 shows the combination of the Bias and TF operators.

Algorithm 7: The Bias and TF operators in the SC2N algorithm.

Input: The updated population of compositions,
The updated weight matrix.

Begin
for i = *1 to* N_{pop} **do**
  if $rand \leq \beta$
  then
    ———————- Bias for new compositions ———————
    % N_b: No. of biased compositions in the population of new compositions
    $N_b = Round(numOfAS \times \beta)$ **for** j = *1:* N_b **do**
      CC^{Input} (i, $Integer\ rand\ [0, numOfAS]) = LB + (UB - LB) \times rand$
    end
    ———————-Bias for updated weight matrix———————
    % N_{wb}: No. of biased compositions in the updated weight matrix
    $N_{wb} = Round(N_{pop} \times \beta)$
    for $j = 1 : N_{wb}$ **do**
      $W^{Updated}$ (j, $Integer\ rand\ [0, Npop]) = U(0, 1)$.
    end
  end
  ———————- Transfer Function (TF) operator ———————
  Apply the (TF) operator using Eq. (3.16).
end
End
Output: The final population of compositions of the current iteration,
(CC^{Best}), (W^{Best}).

Finally, the pseudo-code of the SC2N algorithm is given below.

Algorithm 8: The SC2N algorithm.

Input: The population size (N_{pop});
The maximum number of iterations ($Max_{iteration}$).

Begin
Perform the initial round of the SC2N algorithm using Algorithm 5;
while $Max_{iteration}$ *is not reached* **do**
    Updating the population and the weight matrix using Algorithm 6;
    Perform the Bias and TF operators using Algorithm 7;
    Update the temporal best composition and its corresponding target weight;
    Update the value of β using Eq. (3.10);
End
Output: ($CC^{BestFinal}$);
($W^{BestFinal}$).

3.4 Conclusion

In this chapter, the proposed approach addresses the problem of the QoS-aware services composition using the NNA algorithm. In the next chapter, the performance and effectiveness of the SC2N algorithm will be evaluated and compared with an other approach.

Chapter 4

Evaluation and simulation results

Contents

4.1 Introduction

In this chapter, the performance of the SC2N approach is assessed according to different simulation scenarios of services composition. First, the simulation environment is presented. Then, the scenarios and the dataset used in the evaluation are presented. Finally, the performance of the proposed approach is compared with the performance of another algorithm in terms of composition time and utility value of the composition.

4.2 Scenarios and methodology

4.2.1 Simulation environment

The proposed SC2N approach was implemented and evaluated using Matlab R2020a running upon a machine equipped with a 64-bit Windows OS, an Intel Celeron N2840 CPU with a frequency of 2.16 GHz and 4 GB RAM.

4.2.2 Evaluation scenarios and dataset

The performance evaluation is conducted using the QWS dataset ver 2.0 that includes a set of 2 507 web services described by their QoS values that were obtained from measurements during the year of 2008, where each row in this dataset represents a service and its corresponding eleven measurements. The first nine elements are QoS values attributes while the last two parameters represent the service name and a reference to the Web Services Description Language (WSDL) document [Al-Masri and Mahmoud, 2008]. We opted for the QWS dataset ver 2.0 because it provides a set of QoS attribute values needed for the comparison (response time, throughput, availability, reliability, and cost). The services composition scenarios considered in the simulations are generated by varying the number of abstract services in a services composition from 5 to 20 while the number of concrete services varies from 200 to 1 000 for each abstract service. Each concrete service is described by a vector of five QoS attributes: response time, throughput, availability, reliability, and cost. It should be noted that the composite services considered in the simulation scenarios have a sequential structure.

4.2.3 Performance metrics and simulation parameters

The performances of the SC2N algorithm are compared to the performances of the modified Artificial Bee Colony QoS-aware services Composition Algorithm (mABC-QCA) [Chandra and Niyogi, 2019]; the reason we opted for the mABC algorithm is its performance evaluation that showed that the mABC-QCA algorithm outperforms the five algorithms that it has been compared to, that are: Artificial Bee Colony (ABC) algorithm, Differential Evolution (DE) algorithm, Modified Grey Wolf Optimizer (MGWO) algorithm, Discrete Gbest-guided Artificial Bee Colony (GABC) algorithm and Improved Artificial Bee Colony (IABC) algorithm in terms of composition time, liability and utility of the composition.

The following metrics are considered for the performance evaluation:

Composition time: this metric represents the time required by each services composition algorithm (SC2N and mABC-QCA algorithms) to find the quasi-optimal composition in terms of QoS.

Utility value: this metric represents the utility value in terms of QoS of the best composition obtained by each algorithm (SC2N and mABC-QCA algorithms).

4.3 Simulation results

The performance evaluation of the SC2N and mABC-QCA algorithms is carried out according to two different services composition scenarios with the used parameters' values summarized in Table 4.1

Table 4.1: The simulation parameters.

Parameters	Services composition scenarios	
The population size	30	
The maximum number of iterations	100	
The number of simulations	50	
The number of abstract services	5	5 to 20
The number of concrete services	200 to 1 000	500

4.3.1 Services composition scenario 1

In this scenario, the number of abstract services in the composition is set to 5, while the number of concrete services in each abstract service varies from 200 to 1 000. The purpose of this evaluation scenario is to test the scalability of the proposed algorithms regarding the number of concrete services. The results of the comparison presented below are the average obtained over 50 simulations

4.3.1.1 Composition time vs. The number of concrete services

In this simulation where the number of concrete services is varied, the time required by the SC2N and mABC-QCA algorithms to find the best services composition will be measured. As we can see in Figure 4.1, the composition time of the mABC-QCA algorithm increases excessively with the number of concrete services. Although it outperforms the SC2N algorithm when the number of concrete services is not high, but starting the 720 concrete services the SC2N algorithm requires less composition time than the mABC-QCA algorithm. This is due to the use of the transfer operator that guides the population towards the best composition leading to a low convergence time and consequently to a reduction in composition time.

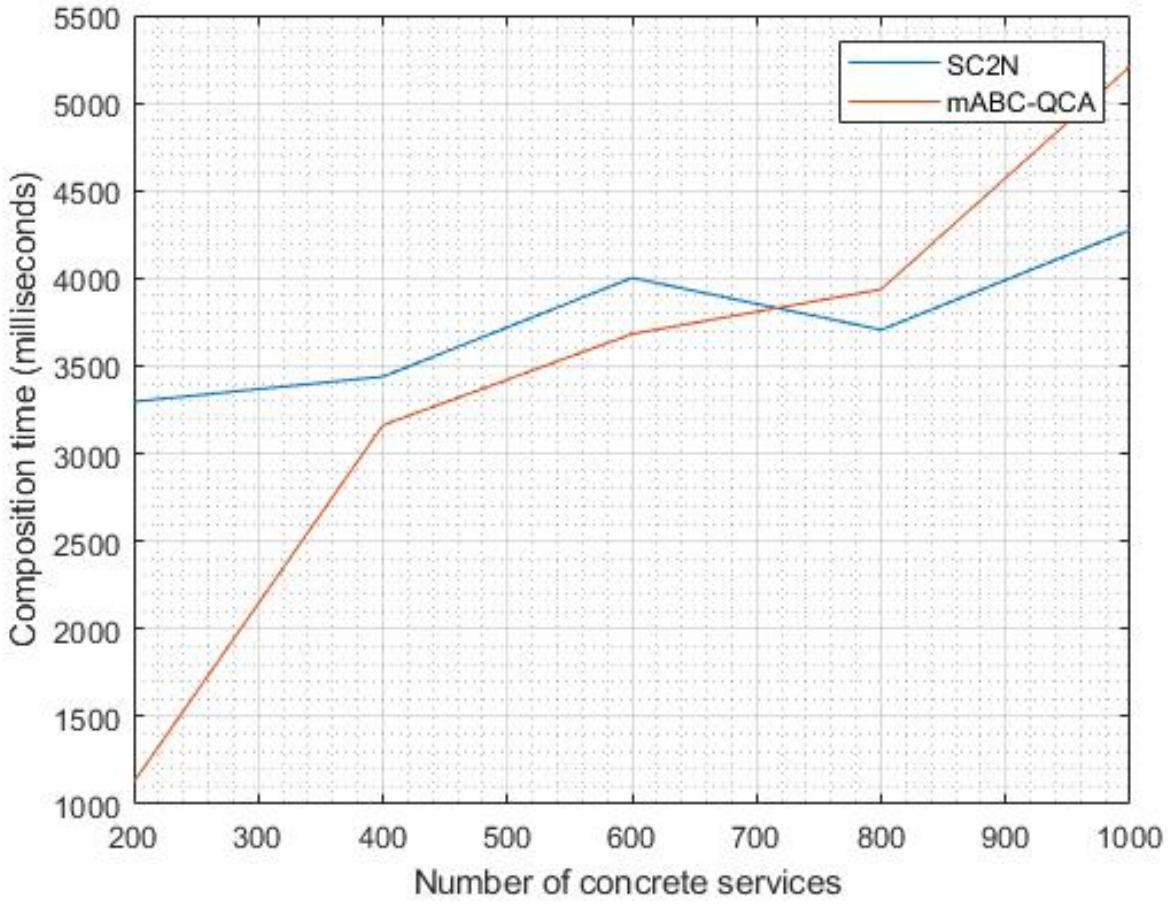

Figure 4.1: Composition time vs. The number of concrete services (scenario 1).

4.3.1.2 The utility value vs. The number of concrete services

This simulation measures and compares the utility value of the compositions in the case of the SC2N and mABC-QCA algorithms. We observe in Figure 4.2 that the utility value of the SC2N algorithm decreases as the number of concrete services increases but still outperforms the mABC-QCA algorithm. This is due to the use of the operators of the SC2N algorithm. Indeed, the Bias operator ensures the diversity of the population while the transfer operator brings the compositions closer to the best composition in terms of QoS.

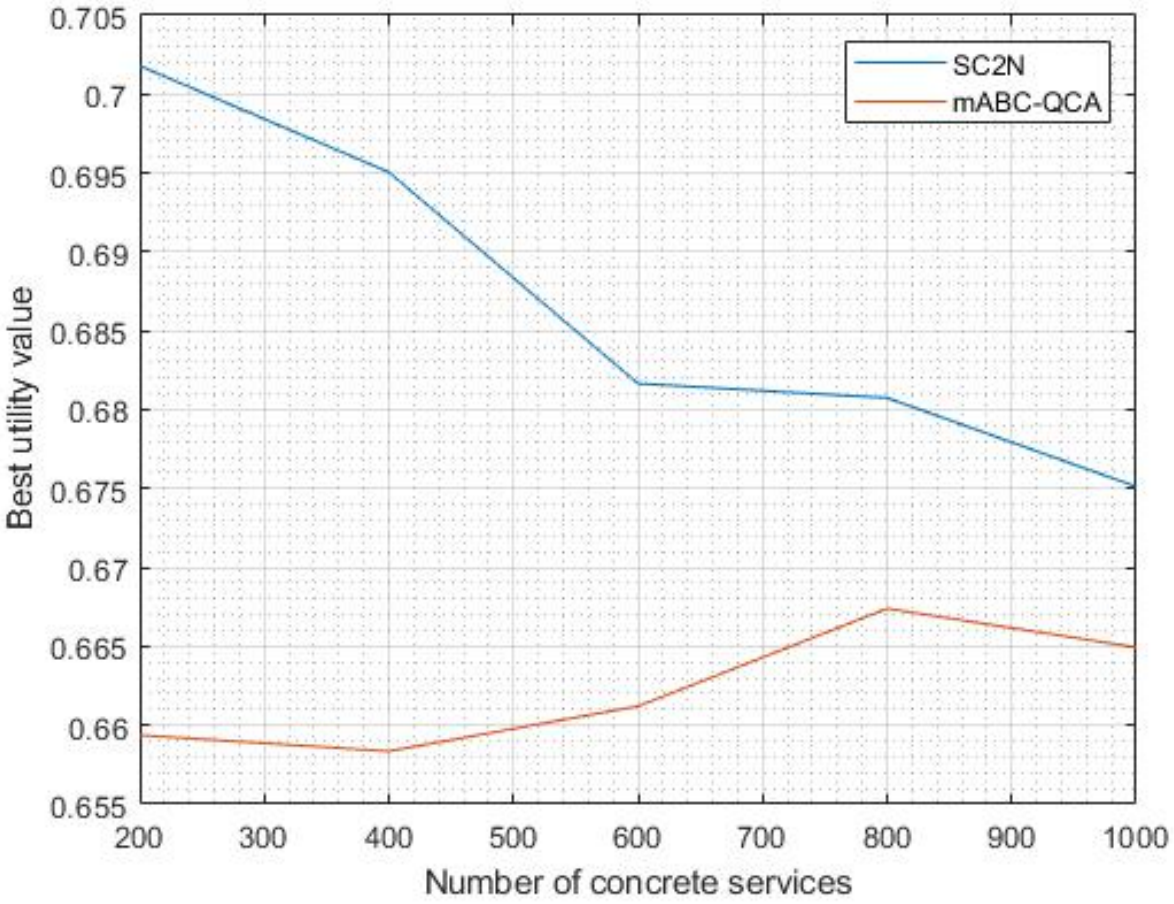

Figure 4.2: The utility value vs. The number of concrete services (scenario 1).

4.3.1.3 The utility value vs. The number of iterations

In this simulation, the utility values in terms of QoS of the compositions are measured through 100 iterations for the SC2N and mABC-QCA algorithms. Figure 4.3 shows that the utility values of the compositions of both algorithms increase with the number of iterations until reaching the highest utility values. This figure also shows that the SC2N algorithm finds a better utility value compared to that obtained in the case of the mABC-QCA algorithm regardless of the number of concrete services. This is due to the used operators in the SC2N algorithm to improve the quality of the compositions.

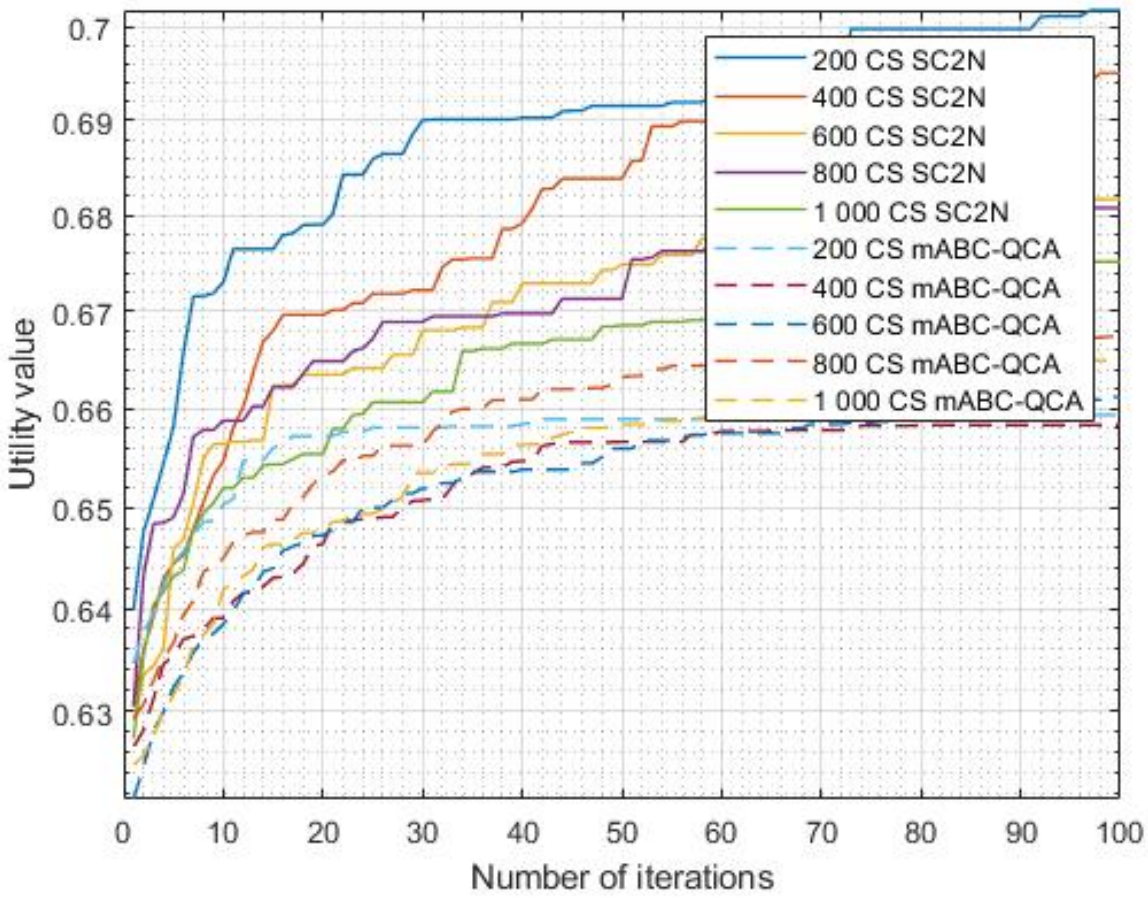

Figure 4.3: The utility value vs. The number of iterations (scenario 1).

4.3.2 Services composition scenario 2

In this scenario, the number of abstract services in the composition varies from 5 to 20, whereas the number of concrete services for each abstract service is set to 500. The results of the comparison presented below are the average obtained over 50 simulations.

4.3.2.1 Composition time vs. The number of abstract services

In this simulation, the composition time taken by the SC2N and mABC-QCA algorithms are compared by varying the number of concrete services. As it can be seen from Figure 4.4, the composition time of both algorithms increases with the number of abstract services, but the SC2N algorithm is significantly more efficient than the mABC-QCA algorithm. This result is due to the fact that the SC2N algorithm converges quickly to the quasi-optimal composition due to the exploitation of the transfer operator.

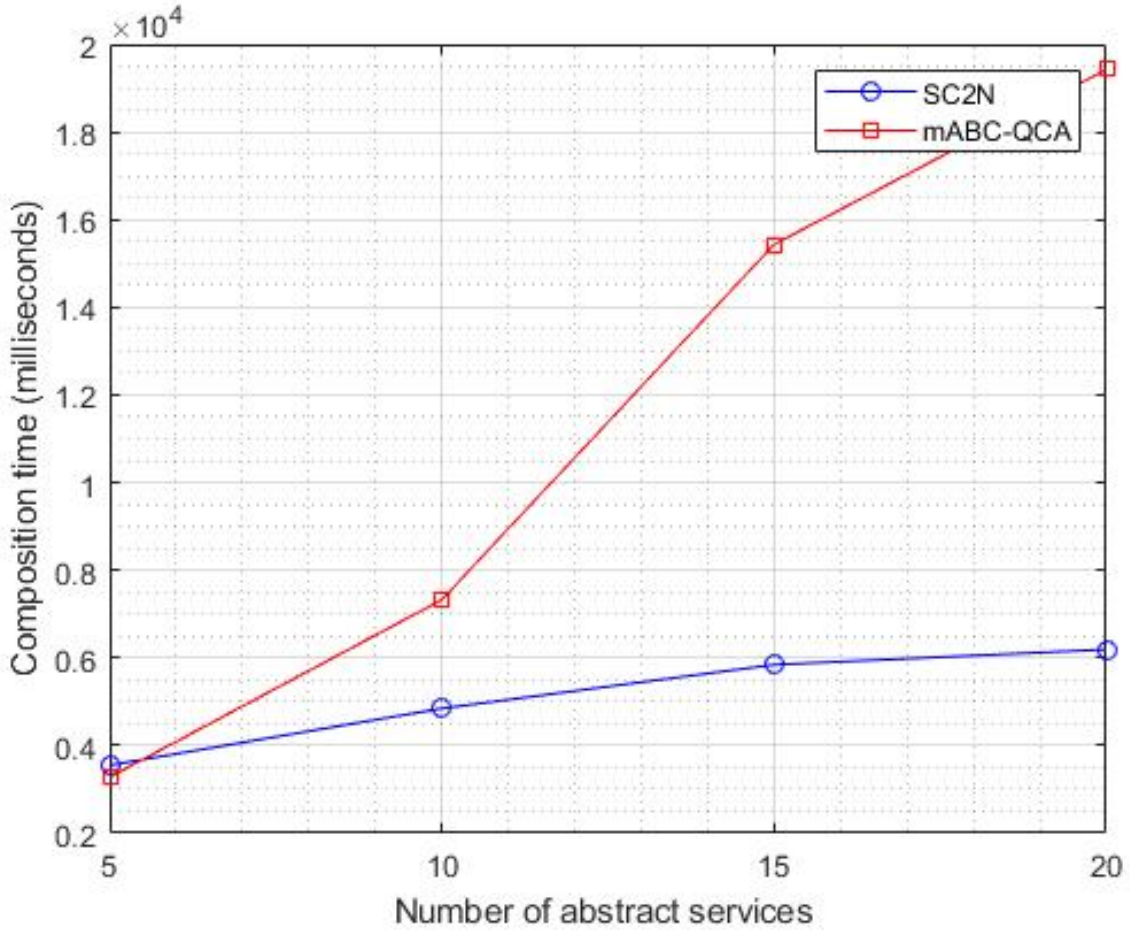

Figure 4.4: Composition time vs. The number of abstract services (scenario 2).

4.3.2.2 The utility value vs. The number of abstract services

This simulation measures and compares the utility value of the compositions in the case of the SC2N and mABC-QCA algorithms. Figure 4.5 shows that the utility value of the two algorithms decreases as the number of abstract services increases, nonetheless the SC2N algorithm provides a better utility value than the mABC-QCA algorithm. This is due to the use of the Bias and transfer operators.

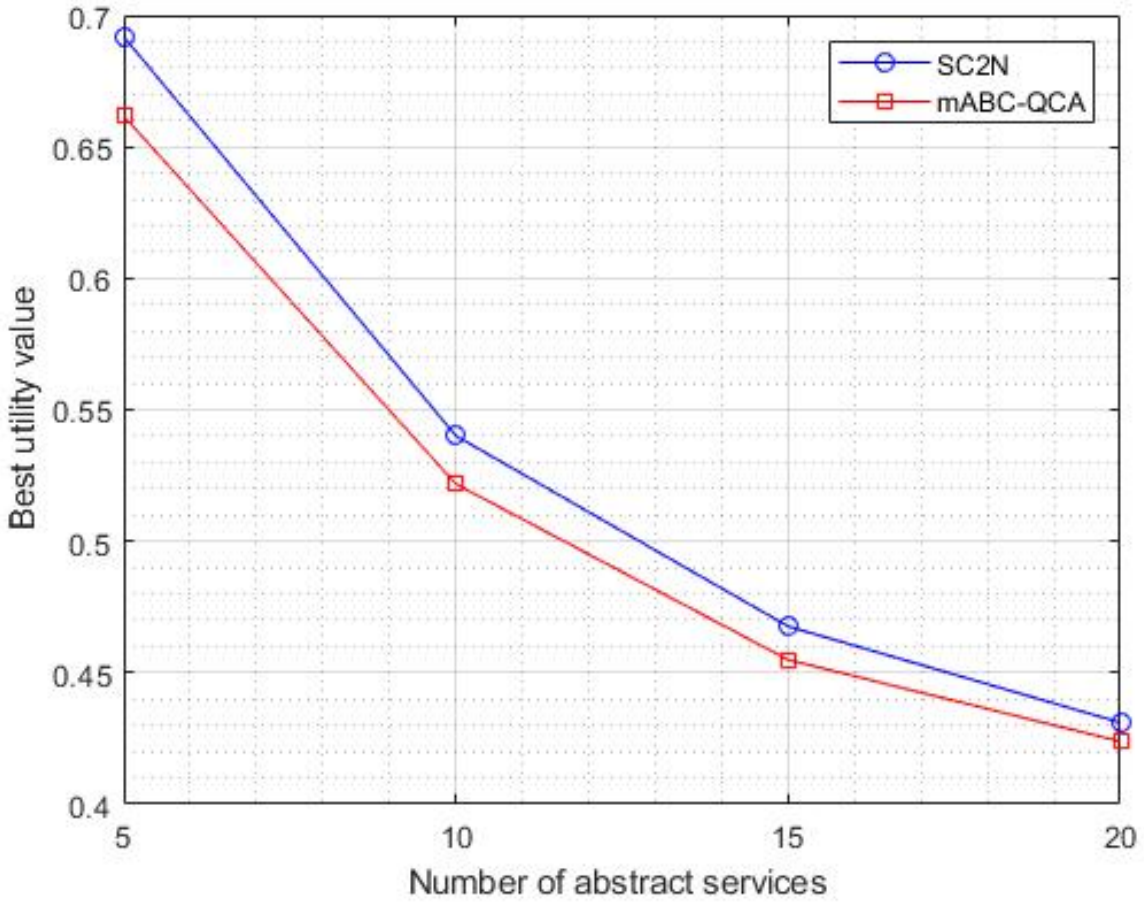

Figure 4.5: The utility value vs. The number of abstract services (scenario 2).

4.3.2.3 The utility value vs. The number of iterations

In this simulation, the utility values of the compositions are measured through 100 iterations for the SC2N and mABC-QCA algorithms. We could see in Figure 4.6 that the utility values of the compositions in both algorithms increase with the increasing number of iterations until reaching the highest utility value then remains stable. However, Figure 4.6 also shows that the SC2N algorithm finds a better utility value compared to the mABC-QCA algorithm regardless of the number of iterations. This result is because the SC2N algorithm allows a better exploration of the search space.

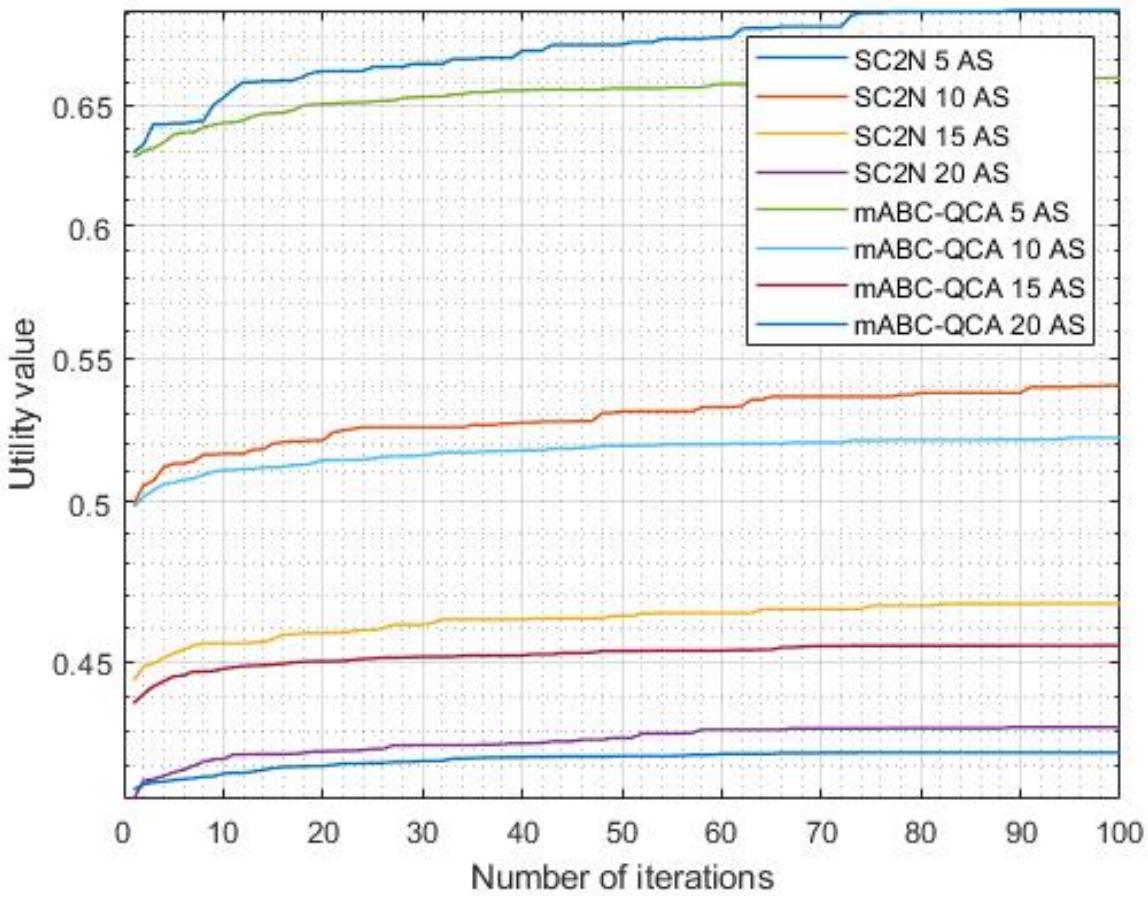

Figure 4.6: The utility value vs. The number of iterations (scenario 2).

4.4 Conclusion

The performance evaluation of SC2N algorithm compared to mABC-QCA algorithm shows that the proposed approach outperforms the mABC-QCA algorithm in terms of composition time and utility of the composition, mainly due to the exploitation of the unique structure of the ANNs and the use of Bais and transfer operators that allow the SC2N algorithm to converge quickly to the best composition.

Conclusion and perspectives

In this thesis, a Services Composition algorithm based on Neural Network (SC2N) is proposed to solve the QoS-aware services composition problem in the context of smart environments. After the random generation of the initial population of compositions, the SC2N algorithm performs in three main different stages. The updating phase helps to obtain new compositions and the Bias phase ensures the diversity of the compositions from one generation to another, whereas the TF operator phase brings the compositions closer to the best composition in terms of the utility value. The comparison results obtained in the simulation scenarios demonstrate that the SC2N algorithm is efficient in terms of composition time and utility thanks to its unique structure and its operators.

In addition, our work opens up other avenues of research where several perspectives and extensions can be proposed such as:

- We are considering the extension of our approach by the addition of the construction of the workflow between services.
- The use of other composition structures: The proposed SC2N approach only deals with the sequential composition and it would be interesting to consider other composition structures such as conditional, loops and parallel.
- The initial population can influence the speed of convergence as well as the quality of the compositions, which is why we consider the use of a population initialization technique instead of a random generation.

Bibliography

[Al-Masri and Mahmoud, 2007a] Al-Masri, E. and Mahmoud, Q. H. (2007a). Discovering the best web service. In *Proceedings of the 16th international conference on World Wide Web*, pages 1257–1258.

[Al-Masri and Mahmoud, 2007b] Al-Masri, E. and Mahmoud, Q. H. (2007b). Qos-based discovery and ranking of web services. In *2007 16th international conference on computer communications and networks*, pages 529–534. IEEE.

[Al-Masri and Mahmoud, 2008] Al-Masri, E. and Mahmoud, Q. H. (2008). Investigating web services on the world wide web. In *Proceedings of the 17th international conference on World Wide Web*, pages 795–804.

[Anthony Steed, 2009] Anthony Steed, M. F. O. (2009). *Networked Graphics*. Morgan Kaufmann.

[Baryannis and Plexousakis, 2010] Baryannis, G. and Plexousakis, D. (2010). *Automated Web Service Composition: State of the Art and Research Challenges*. ICS-FORTH.

[Bekkouche, 2018] Bekkouche, A. (2018). *Vers une composition automatique des services web.* PhD thesis, Abou Bekr Belkaid Tlemcen UNIVERSITY.

[Bekkouche et al., 2017] Bekkouche, A., Benslimane, S. M., Huchard, M., Tibermacine, C., Hadjila, F., and Merzoug, M. (2017). Qos-aware optimal and automated semantic web service composition with user's constraints. *Service Oriented Computing and Applications*, 11(2):183–201.

[Chandra and Niyogi, 2019] Chandra, M. and Niyogi, R. (2019). Web service selection using modified artificial bee colony algorithm. *IEEE Access*, 7:88673–88684.

[Dahan et al., 2019] Dahan, F., Mathkour, H., and Arafah, M. (2019). Two-step artificial bee colony algorithm enhancement for qos-aware web service selection problem. *IEEE Access*, 7:21787–21794.

[Elmaghraoui et al., 2017] Elmaghraoui, H., Benhlima, L., and Dalila, C. (2017). Dynamic web service composition using and/or directed graph. pages 1–8.

[Fan et al., 2019] Fan, G., Zhu, M., and Cui, X. (2019). Optimizing web service composition with graphplan and fuzzy control. *Journal of Ubiquitous Systems & Pervasive Networks*, 11(2):15–21.

[Fki et al., 2017] Fki, E., Tazi, S., and Drira, K. (2017). Automated and flexible composition based on abstract services for a better adaptation to user intentions. *Future Generation Computer Systems*, 68:376–390.

[Guelzim et al., 2016] Guelzim, T., Obaidat, M., and Sadoun, B. (2016). Introduction and overview of key enabling technologies for smart cities and homes. In *Smart cities and homes*, pages 1–16. Elsevier.

[Khanouche et al., 2019a] Khanouche, M. E., Atmani, N., and Cherifi, A. (2019a). Improved teaching learning-based qos-aware services composition for internet of things.

[Khanouche et al., 2019b] Khanouche, M. E., Atmani, N., Cherifi, A., Chibani, A., Matson, E. T., and Amirat, Y. (2019b). Qos-aware agent capabilities composition in harms multi-agent systems. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 127–138. Springer.

[Labbaci et al., 2017] Labbaci, H., Medjahed, B., and Aklouf, Y. (2017). A deep learning approach for long term qos-compliant service composition. In *International Conference on Service-Oriented Computing*, pages 287–294. Springer.

[Li et al., 2018a] Li, C., Guan, J., Liu, T., Ma, N., and Zhang, J. (2018a). An autonomy-oriented method for service composition and optimal selection in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 96(5-8):2583–2604.

[Li et al., 2018b] Li, F., Zhang, L., Liu, Y., and Laili, Y. (2018b). Qos-aware service composition in cloud manufacturing: a gale-shapley algorithm-based approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

[Li et al., 2017] Li, F., Zhang, L., Liu, Y., Laili, Y., and Tao, F. (2017). A clustering network-based approach to service composition in cloud manufacturing. *International Journal of Computer Integrated Manufacturing*, 30(12):1331–1342.

[Liew, 2016] Liew, S. S. (2016). *An Efficient and Effective Convolutional Neural Network for Visual Pattern Recognition*. PhD thesis.

[Liu et al., 2019] Liu, J.-W., Hu, L.-Q., Cai, Z.-Q., Xing, L.-N., and Tan, X. (2019). Large-scale and adaptive service composition based on deep reinforcement learning. *Journal of Visual Communication and Image Representation*, 65:102687.

[Mabrouk et al., 2009] Mabrouk, N. B., Beauche, S., Kuznetsova, E., Georgantas, N., and Issarny, V. (2009). Qos-aware service composition in dynamic service oriented environments. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 123–142. Springer.

[Marquardt and Uhrmacher, 2008] Marquardt, F. and Uhrmacher, A. M. (2008). Evaluating ai planning for service composition in smart environments. In *Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia*, pages 48–55.

[Nazerfard and Cook, 2012] Nazerfard, E. and Cook, D. J. (2012). Bayesian networks structure learning for activity prediction in smart homes. In *2012 Eighth International Conference on Intelligent Environments*, pages 50–56. IEEE.

[Rodriguez-Mier et al., 2015] Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2015). Hybrid optimization algorithm for large-scale qos-aware service composition. *IEEE transactions on services computing*, 10(4):547–559.

[Sadollah et al., 2018] Sadollah, A., Sayyaadi, H., and Yadav, A. (2018). A dynamic metaheuristic optimization model inspired by biological nervous systems: Neural network algorithm. *Applied Soft Computing*, 71:747–782.

[Seghir and Khababa, 2018] Seghir, F. and Khababa, A. (2018). A hybrid approach using genetic and fruit fly optimization algorithms for qos-aware cloud service composition. *Journal of Intelligent Manufacturing*, 29(8):1773–1792.

[Yang et al., 2019] Yang, Y., Ke, W., Wang, W., and Zhao, Y. (2019). Deep learning for web services classification. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 440–442. IEEE.

[Zeng et al., 2003] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421.

RÉSUMÉ

L'environnement intelligent est un écosystème où les utilisateurs interagissent en permanence avec des objets intelligents (capteurs, smartphones, appareils, etc.) pour améliorer leur quotidien. Ce type d'intelligence nécessite la coopération de plusieurs appareils intelligents offrant différents services pour satisfaire la demande croissante de l'utilisateur, cependant, les exigences de l'utilisateur dépassent souvent un seul service ce qui explique la forte demande de composition. Avec la croissance rapide du nombre de services fonctionnellement équivalents, la qualité de service (QoS) est devenue un facteur important pour distinguer les services fonctionnellement équivalents. Cependant, le défi de sélectionner le meilleur ensemble de services à composer en tenant compte des contraintes globales de QoS imposées par l'utilisateur est devenu un problème NP-difficile. Dans cette thèse, nous proposons (SC2N) un algorithme de composition de services basé sur le réseau neuronal pour résoudre le problème de composition de services sensible à la qualité de service dans le contexte d'environnements intelligents. Les résultats de comparaison obtenus dans les scénarios de simulation démontrent que l'algorithme SC2N est efficace en termes de temps de composition et d'utilité grâce à sa structure unique et ses opérateurs; la phase Bias assure la diversité des compositions d'une génération à l'autre, tandis que la phase opérateur TF rapproche les compositions de la meilleure composition en termes de valeur d'utilité.

Mots clés: Environnements intelligents, composition de services, qualité de service (QoS), contraintes globales, algorithme méta-heuristique, algorithme de réseau neuronal (NNA), algorithme de composition de services basé sur le réseau neuronal (SC2N).

ABSTRACT

The smart environment is an ecosystem where users constantly interact with smart objects (sensors, smartphones, devices, etc.) to improve their daily lives. This type of intelligence requires the cooperation of several smart devices offering different services to satisfy the increasing demand of the user, however, the user's requirements often exceed a single service which explains the great demand for the composition. With the rapid growth in the number of functionally equivalent services, Quality of Service (QoS) has become an important factor in distinguishing between functionally equivalent services. However, the challenge of selecting the best set of services to compose taking into account the global QoS constraints imposed by the user has become an NP-hard problem. In this thesis, we propose the Services Composition algorithm based on Neural Network (SC2N) to solve the QoS-aware services composition problem in the context of smart environments. The comparison results obtained in the simulation scenarios demonstrate that the SC2N algorithm is efficient in terms of composition time and utility thanks to its unique structure and its operators; the Bias phase ensures the diversity of the compositions from one generation to another, whereas the TF operator phase brings the compositions closer to the best composition in terms of the utility value.

Keywords: Smart environments, services composition, Quality Of Service (QoS), global constraints, meta-heuristic algorithm, Neural Network Algorithm (NNA), Services Composition algorithm based on Neural Network (SC2N).