Genetic Programming for QoS-Aware Data-Intensive Web Service Composition and Execution

by

Yang Yu

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Engineering
in Software Engineering.

Victoria University of Wellington
2015

Abstract

Web service composition has become a promising technique to build powerful enterprise applications by making use of distributed services with different functions. In the age of big data, more and more web services are created to deal with a large amount of data, which are called data-intensive services. Due to the explosion in the volume of data, providing efficient approaches to composing data-intensive services will become more and more important in the field of service-oriented computing. Meanwhile, as numerous web services have been emerging to offer identical or similar functionality on the Internet, web service composition is usually performed with end-to-end Quality of Service (QoS) properties which are adopted to describe the non-functional properties (e.g., response time, execution cost, reliability, etc.) of a web service. In addition, the executions of composite web services are typically coordinated by a centralized workflow engine. As a result, the centralized execution paradigm suffers from inefficient communication and a single point of failure. This is particularly problematic in the context of data-intensive processes. To that end, more decentralized and flexible execution paradigms are required for the execution of data-intensive applications.

From a computational point of view, the problems of QoS-aware data-intensive web service composition and execution can be characterised as complex, large-scale, constrained and multi-objective optimization problems. Therefore, genetic programming (GP) based solutions are presented in this thesis to address the problems. A series of simulation experiments are provided to demonstrate the performance of the proposed approaches, and the empirical observations are also described in this thesis.

Firstly, we propose a hybrid approach that integrates the local search procedure of tabu search into the global search process of GP to solving the problem of QoS-aware data-intensive web service composition. A mathematical model is developed for considering the mass data transmission across different component services in a data-intensive service composition. The experimental results show that our proposed approach can provide better performance than the standard GP approach and two traditional optimization methods.

Next, a many-objective evolutionary approach is proposed for tackling the QoS-aware data-intensive service composition problem having more than three competing quality objectives. In this approach, the original search space of the problem is reduced before a recently developed many-objective optimization algorithm, NSGA-III, is adopted to solve the many-objective optimization problem. The experimental results demonstrate the effectiveness of our approach, as well as its superiority than existing single-objective and multi-objective approaches.

Finally, a GP-based approach to partitioning a composite data-intensive service for decentralized execution is put forth in this thesis. Similar to the first problem, a mathematical model is developed for estimating the communication overhead inside a partition and across the partitions. The data and control dependencies in the original composite web service can be properly preserved in the deployment topology generated by our approach. Compared with two existing heuristic algorithms, the proposed approach exhibits better scalability and it is more suitable for large-scale partitioning problems.

Acknowledgments

This master thesis would not have been possible without the kind support and help of many people. First and foremost, I would like to express my sincere gratitude to my supervisors, Dr. Hui Ma and Prof. Mengjie Zhang who were abundantly helpful and offered invaluable assistance, support and guidance throughout my thesis. I also would like to express my love and gratitude to my beloved families for their understanding and endless love, through the duration of my studies.

Contents

Chapter 1

Introduction

Service-oriented architecture (SOA) [63] is a widely accepted and engaged paradigm for the realization of complex business processes. The aim of SOA is to implement business processes covering different organisations and computing platforms in a dynamic and loosely-coupled manner. As a promising technology to implement such a service-oriented architecture, web services encapsulate software functions and make them available to anyone in the world over the network via standard interfaces and protocols (e.g., SOAP and WSDL). The advent of web services has boosted the creation of business applications by reusing existing resources on the network, rather than building new applications from scratch to fulfill business functional requirements.

Web services are self-contained and self-describing modular application components deployed on the Internet, which follow certain technical standards such as Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI) [78]. In recent years, many software functions have been published over the Internet or the Cloud in the format of web services, which enable better integration, scalability and availability for business. As a distributed computing technology, web services allow their functionalities to be available to anyone in the world. For example, the

information about weather in New Zealand could be published through a web service that, given a ZIP code, will provide the weather information for that ZIP code. A sketchy example of how a web service works is illustrated in Figure 1.1.

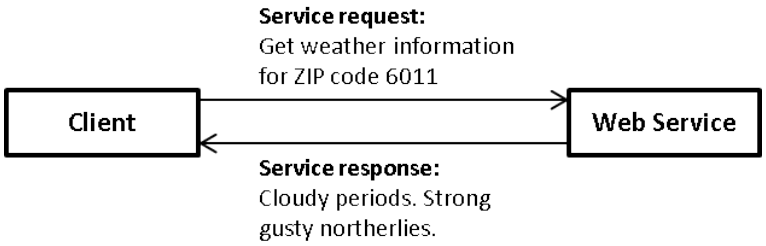


Figure 1.1: An example of a weather information web service

However, a web service itself has a limited functionality and when no single web service is able to respond to the user's request, it is necessary to compose a range of existing services together in order to provide new value-added and complex functionality, which is referred to as *web service composition*, and the aggregated web service becomes a *composite web service*. A typical scenario of web service composition is an online travel agent shown in Figure 1.2 that is composed of three tasks for flight booking, hotel reservation and car rental. In other terms, the user needs to execute each of these corresponding web services manually and these tasks can be time and effort consuming. For that reason, how to compose a number of web services automatically without user interference or intervention attracts a lot of interest. On the other hand, today, a large number of web services on the Internet offer identical or overlapping functionality but present various non-functional characteristics which are called *quality of service (QoS)* properties such as response time, execution cost and reliability. Therefore, how to select a suitable web service that satisfies user's requirements still remains an open question, and the non-functional QoS factors become significant criteria that need to be considered in web service composition. To select web services among a great number of candi-

date services to meet global end-to-end QoS requirements while fulfilling the functional requirements is called the problem of *QoS-aware web service composition*.

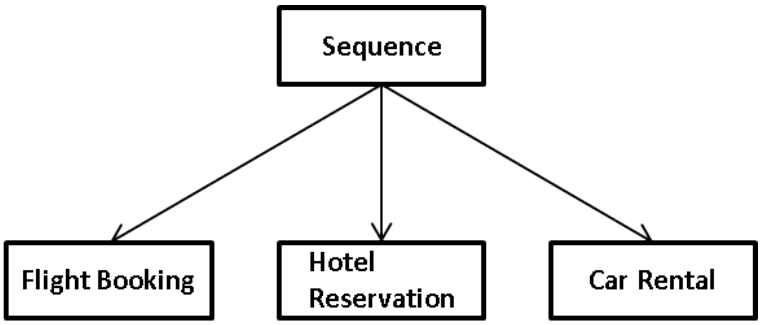


Figure 1.2: An example of an online travel agent system

Aside from business processes, the service-oriented approach using web services is also of great interest for the implementation of data intensive processes such as data mining and image processing. Such web services are defined as *data-intensive services* that generally have large amounts of data as their inputs and outputs [22]. For example, as depicted in Figure 1.3, a facial recognition solution for video and image content could be published through a web service that, given a set of original images, will provide a set of processed images with identified facial features. Over the recent years, the amount of data generated by humanities, scientific activities, as well as commercial applications from a diverse range of fields has been increasing exponentially. Data volumes used in the fields of sciences and engineering, finance, media, online information resources and so on, are expected to double every two years over the next decade and further [84]. There is no doubt in the industry and research community that the importance of data-intensive computing has been raising and will continue to be the foremost research field. As a result, data-intensive services based applications have become the most challenging type of applications in SOA. Also, data-intensive service composition has become an appealing

research area in academia and industry.

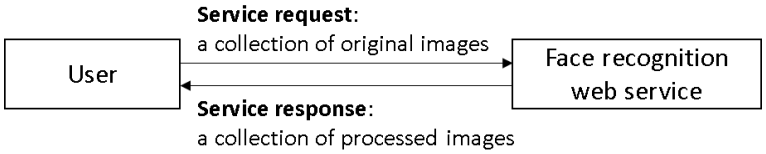


Figure 1.3: An example of a face recognition web service

1.1 Research Problems

1.1.1 Motivations

With the increasing presence and adoption of web services on the World Wide Web, the composition strategy of a variety of services provides maximal flexibility when designing and implementing data-intensive applications. On the other hand, with the tremendous growth of identical or similar services, QoS is usually employed to describe the non-functional characteristics of web services. For traditional web service compositions, the time and cost of data transmission among component services are negligible compared with the execution time of component services. However, as a considerable amount of data needs to be exchanged between the components in a composite data-intensive service, the movement of mass data has a great influence on the overall performance of the composite web service. Therefore, it becomes inevitable to take into account non-functional properties (e.g., response time, execution cost, reliability, etc.) during the dynamic process of data-intensive service composition, especially attention should be drawn to the transfer time and transfer cost of mass data transmission between collaborating web services.

A survey on QoS-aware web service composition reveals that most research on automated or semi-automated service composition falls in realm

of single-objective optimization. In this approach, different performance objectives are combined into a single objective according to a certain unity function then a particular optimization technique such as genetic algorithm is applied to optimize the objective. In the situation where the user's preference information is completely unknown, the methods in such a category are difficult to apply because each quality dimension needs to be assigned a weight which is not well known *a priori*. Also different dimensional qualities may conflict with one another in the real world. For example, quicker response and lower price are highly demanded, but in practice they are typically in conflict. As more quality dimensions are constrained by the user, the aforementioned problem becomes more complicated. Therefore, it is unlikely to find an optimal solution in one dimension of the objectives without causing unnecessary suffering to another. This is referred to as the problem of *many-objective QoS-aware web service composition* in this work. However, limited work has focused on this problem for data-intensive service compositions when more than three quality dimensions are considered simultaneously.

Typically, a composite web service is executed by a single coordinator node which receives service user's request, makes necessary data transformations and invokes each component service. As the name implies, the coordinator node is responsible for the coordination of all the data and control flow between the component services, and hence becomes a potential performance bottleneck. This execution mode of composite web services is referred to as *centralized orchestration* [14]. As depicted in Figure 1.4, all data is transferred between various component services via the coordinator node instead of being transferred directly from the point of generation to the point of consumption, which may cause unnecessary network traffic overloading. In addition, it is possible for a component service to produce a lot of data that is irrelevant to the desired function of the composite web service. However, the data will be still transferred to the coordinate node, thereby resulting in increased workload on the network

as well as the increase of response time. This is particularly problematic for data-intensive service compositions in which huge collections of data are transferred.

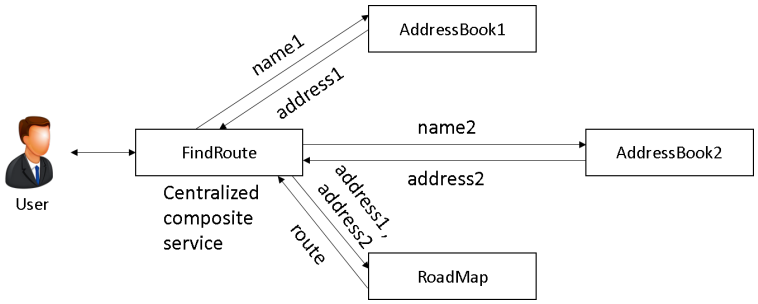


Figure 1.4: An example of centralized orchestration

From a computational point of view, the problems of QoS-aware data-intensive web service composition and execution have proven to be NP-hard. Genetic programming (GP) [48] which is inspired by the process of natural evolution has been successfully applied in a wide range of problem domains [48] such as building electronic circuits or control systems to solve complex, large-scale, constrained and multi-objective optimization problems. The past successful applications of GP motivate our selection of GP to address the problems presented in this thesis, which are also characterized as complex, large-scale, constrained and multi-objective optimization problems.

1.1.2 Objectives

Based on the above motivational discussion, the overall objective of this thesis is to develop efficient and scalable GP-based approaches to finding a data-intensive web service composition and realizing decentralized execution of the composite web service found, with the ability to deal with many and often conflicting quality dimensions. To be specific, this objective is decomposed into three separate objectives that are to be addressed

in this thesis, and they are described as follows.

- determining how to efficiently compose a set of data-intensive services by considering the non-functional attributes of the composite web service, especially paying attention to the time and cost generated by mass data access and transfer,
- solving the problem of QoS-aware data-intensive service composition for the situation where users are not certain about the importance of each quality criterion, and many and often conflicting quality objectives are considered,
- coping with the poor scalability and performance degradation such as unnecessary network traffic overloading and increased response time raised by centralized orchestration.

1.2 Major Contributions

In the field of QoS-aware data-intensive web service composition and execution, the key contributions of this thesis are three-fold. Firstly, the data transmission among the components in data-intensive service compositions is taken into consideration when selecting and composing a number of data-intensive web services. So far, the existing literature has neglected the time and cost spent on mass data transmission between data-intensive web services. As data transmission has a great influence on the overall performance of data-intensive service compositions, approaches without considering it cannot satisfy users' non-functional requirements accurately. Our work overcomes this issue by proposing a mathematical model to measure the communication time and cost between the components in a composite data-intensive service, and a hybrid GP-Tabu approach is proposed to address the service composition problem. The simulation experiments demonstrate that the hybrid approach outperforms GP, tabu search

and integer linear programming in finding more satisfying data-intensive service compositions. A research paper [93] containing this work has been published in the Tenth International Conference on Simulated Evolution And Learning (SEAL 2014).

Secondly, we propose a many-objective evolutionary approach named F-MOGP to solving the service composition problem when preference information is not well known a priori, which fills the gap in the literature on QoS-aware data-intensive web service composition depending on evolutionary many-objective optimization algorithms. The two-phase F-MOGP approach starts with a search space reduction strategy in order to reduce the huge search space of a complex problem. The experimental results show that the search space can be significantly reduced by the proposed search space reduction algorithm, which leads to a dramatic reduction of the computation time required by the optimization phase. By performing a comparative study of the performance of two existing EMOs (i.e., NSGA-II and SPEA2) based on large datasets, we reach the conclusion that our approach demonstrates better performance when more than three quality objectives are considered in the search process for a data-intensive service composition. A research paper [94] about this contribution has been published in the 2015 IEEE Congress on Evolutionary Computation (CEC2015).

The third contribution of this thesis is a GP-based approach that is proposed to split a data-intensive BPEL process into a set of sub-processes for decentralized execution of composite data-intensive web services. The partitioning process takes the data communication overhead occurred inside a partition and across the partitions into consideration. Compared with two state-of-art methods which are MDU [61] and PGM [61] respectively, the proposed approach is scalable with respect to the size of data-intensive service compositions. In other terms, for large-scale partitioning problems, our approach requires less computation time to generate higher-quality partitioning execution plans compared with the two exist-

ing methods. A research paper has been submitted to the IEEE International Conference on Web Services (ICWS).

1.3 Organisations

The remainder of this thesis is organized as follows. The background of our research problems and an overview of recent research efforts are presented in the following chapter. Chapter 3 gives a description of our hybrid GP-Tabu approach to QoS-aware data-intensive web service composition, meanwhile a series of experiments are conducted to evaluate the effectiveness and efficiency of the proposed approach. In Chapter 4, the service composition problem is solved using a many-objective evolutionary approach named F-MOGP, and simulation experiments are performed to evaluate and compare the performance of the approach and two popular EMOs. Chapter 5 presents a GP-based approach for distributed execution of composite data-intensive web services and a set of experiments demonstrate that the proposed approach has better scalability than two existing heuristic algorithms. At the end of the thesis, conclusions are made for our research work, and future perspectives are described.

Chapter 2

Literature Survey

2.1 Background

In recent years, web services that encapsulate the functions of application components have been rapidly developed and play an increasingly important role in implementing a service-oriented architecture. To make full use of single web services which provide limited functions only, *web service composition* that aims to integrate multiple services into workflows in order to supply value-added and complex functions, has become a popular research focus. In web service composition, functional requirements are concerned with the functionalities of a composite web service, e.g., given a flight booking task, the functional requirement is to purchase an airline ticket based on the information provided by a user. In addition to functional capabilities of web services, non-functional requirements (e.g., response time, throughout, availability) have to be matched to maximize user satisfaction expressed as many objectives over a variety of QoS attributes of services. An example of non-functional requirements for the flight booking task is that the composite web service will respond within a certain time period and charges less than a specific amount of money with a preferred level of accuracy and reliability. Nowadays, due to the explosive increase of web services that provide identical or overlapping

functionality on the Web, quality of service namely QoS becomes a key factor in distinguishing these functionally equivalent services.

2.1.1 Atomic Web Service *vs.* Composite Web Service

A *composite web service* is made up of a collection of single web services each of which is referred to as *atomic web service*. The goal of a service composition is to generate the desired outputs given a set of available inputs that are described by semantic concepts. Assume the task scenario is to search for an appropriate flight as well as the weather forecast for the destination based upon the given departure date, return date, departure city and arrival city. In other terms, a web service is requested to take {*departure date, return date, departure city, arrival city*} as inputs in order to produce {*flight information, weather forecast*} as outputs. However, an atomic web service itself has limited functionality which is not sufficient to respond to the user's request. A composite web service that consists of multiple atomic web services is needed to accomplish the task. A valid service composition must guarantee that the inputs of any atomic web service are available either from the outputs of its ancestors or from the original inputs (i.e., from the user).

2.1.2 Quality of Service

QoS can be characterized according to various non-functional properties of web services called *QoS attributes* such as response time, execution cost, availability and reliability. Based on a selection of relevant characteristics in the field of web services, the QoS attributes considered in this research work are latency, execution cost, availability and accuracy all of which are defined as follows.

- $latency\ L$ measures the expected delay in seconds between the moment when a request is sent and the moment when the results are received.

- $execution\ cost\ C$ is the amount of money that a service requester has to pay for executing the web service.

- $availability\ A$ is the probability that a web service is accessible.

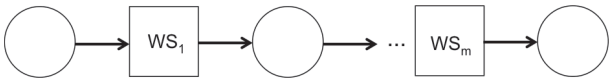
- $accuracy\ R$ is the measurement of the degree to which the real results produced by the web service match the desired results.

Amongst the above four QoS attributes, latency and execution cost are decreasing measures with respect to QoS. In other terms, the grades of these measures increase as their values decrease. In contrast, availability and accuracy are increasing measures of which the grades decrease as their values decrease.

The four basic control structures shared by web service composition languages such as OWL-S and BPEL4WS are $sequence$, $parallel$ ($flow$), $choice$ ($switch$) and $loop$. In the figures below, each rectangle denotes a component service that defines either a composite service or an atomic service, and the circles on the left and right represent its functional inputs and outputs, respectively. The workflow structures and the aggregation functions are described in the following subsections. Note that all the aggregation functions are recursive.

Sequence Construct

For a sequence workflow of tasks, as shown in Figure 2.1, a number of component services (WS_1, ..., WS_n, ..., WS_m) are executed in a sequential order. The outputs of a component service are part of the inputs of its subsequent service. The aggregation functions of latency L and execution cost C are additive, while the availability A and accuracy R functions are multiplicative.

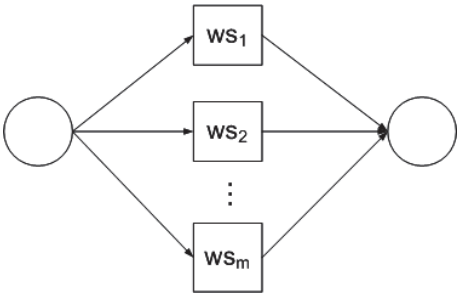


$$L = \sum_{n=1}^{m} l_n, \qquad C = \sum_{n=1}^{m} c_n, \qquad A = \prod_{n=1}^{m} a_n, \qquad R = \prod_{n=1}^{m} r_n$$

Figure 2.1: Sequence Construct

Parallel Construct

As shown in Figure 2.2, two or more paths are executed in parallel to produce different outputs. Except that for latency L, where the aggregated value is the maximum of the latency of component services, the aggregation functions for the other QoS attributes (i.e., execution cost, availability and accuracy) are the same as the ones in the sequence structure.



$$L = MAX\left\{l_n | n \in \{1, \ldots, m\}\right\}$$

$$C = \sum_{n=1}^{m} c_n, \qquad A = \prod_{n=1}^{m} a_n, \qquad R = \prod_{n=1}^{m} r_n$$

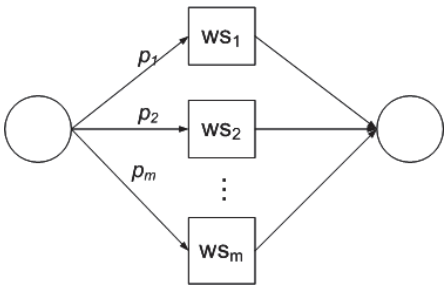
Figure 2.2: Parallel Construct

Choice Construct

Figure 2.3 shows a choice workflow of tasks, where the same outputs can be obtained through multiple different paths, but only one path will selected and executed. Assume the choice structure has m branches, and the percentage for each branch to be selected is p_1,...,p_m, where

$$\sum_{n=1}^{m} p_n = 1.$$

Therefore, all QoS attributes are evaluated as a sum of the multiplication of the attribute value of each component service and its corresponding percentage.



$$L = \sum_{n=1}^{m} p_n * l_n, \qquad C = \sum_{n=1}^{m} p_n * c_n$$

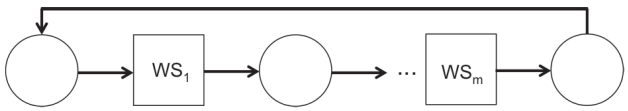
$$A = \sum_{n=1}^{m} p_n * a_n, \qquad R = \sum_{n=1}^{m} p_n * r_n$$

Figure 2.3: Choice Construct

Loop Construct

In Figure 2.4 that shows a loop construct, one or more component services are executed repeatedly until a given condition is verified. Assume the number of iterations is k, then the aggregation functions for latency L and

execution cost C are $t \cdot k$ and $c \cdot k$ respectively. For availability A and accuracy R, the aggregation functions are the kth power of the value of one iteration, i.e., a^k and r^k. Here, l, c, a and r denote the latency, execution cost, availability and accuracy, respectively, of a composite service of each cycle within the loop.



$$T = k * t, \qquad C = k * c, \qquad A = a^k, \qquad R = r^k$$

Figure 2.4: Loop Construct

QoS reflects the non-functional properties of web services which in turn have an influence on user satisfaction, thereby the QoS factors have become significant criteria that cannot be overlooked in web service composition. The problem of *QoS-aware service composition* denotes selecting a number of atomic web services while obtaining the highest possible QoS of the service composition and satisfying the global constraints posed by the user. Here, global constraints define an upper or lower bound for the aggregated QoS values of a composite service. To optimize the quality of a service composition, a method must be applied to estimate the QoS of the service composition from its constituent services. This estimation is called QoS aggregation, and the QoS aggregation formulas defined for the above four basic composition patterns and major QoS attributes are summarized in Table 2.1.

2.1.3 Single-Objective, Multi-Objective and Many-Objective Optimization

Single-objective optimization refers to the optimization problems in which there is a single objective function. In practical problems, more than one

Table 2.1: Aggregation formulae for each pair QoS attribute - workflow structure

QoS attribute	Sequence	Parallel (Flow)	Choice (Switch)	Loop
Latency	$L = \sum_{i=1}^{j} l_i$	$L = MAX\{l_i \mid i \in \{1, \ldots, j\}\}$	$L = \sum_{i=1}^{j} p_i * l_i$	$L = k * l$
Execution cost	$C = \sum_{i=1}^{j} c_i$	$C = \sum_{i=1}^{j} c_i$	$C = \sum_{i=1}^{j} p_i * c_i$	$C = k * c$
Availability	$A = \prod_{i=1}^{j} a_i$	$A = \prod_{i=1}^{j} a_i$	$A = \sum_{i=1}^{j} p_i * a_i$	$A = a^k$
Accuracy	$R = \prod_{i=1}^{j} r_i$	$R = \prod_{i=1}^{j} r_i$	$R = \sum_{i=1}^{j} p_i * r_i$	$R = r^k$

objective is naturally involved, and there does not exist a single solution that simultaneously optimizes each objective. In that case, *multi-objective optimization* is performed to find a number of Pareto-optimal (i.e., non-dominated) solutions. Many evolutionary multi-objective optimization (EMO) algorithms such as NSGA-II [19] and SPEA2 [103], have been proposed and successfully applied for tackling two-objective and three-objective problems. However, it has been widely accepted that these existing EMO algorithms may lose their effectiveness for *many-objective optimization* problems in which more than three objectives arise. A few many-objective optimization algorithms have been introduced to overcome the known disadvantages of EMO algorithms until recently.

Pareto Dominance

A multi-objective search space is partially ordered in the sense that there is no longer a strict linear ordering of solutions. Instead, two arbitrary solutions are related to each other in two possible ways: either one dominates the other or neither dominates. Individual A Pareto dominates individual B if and only if A is at least as good as B in all objectives, and is superior to B in at least on objective. In Figure 2.5, for example, individual A dominates (is better than) individual B along the y axis, but B dominates A along the x axis. Thus there is no simple ordering between them. The individual marked 2, however dominates B on both axes and would thus be considered strictly better than B. All individuals that are not dominated by any other individual of a given set are called $nondominated$ regarding

this set.

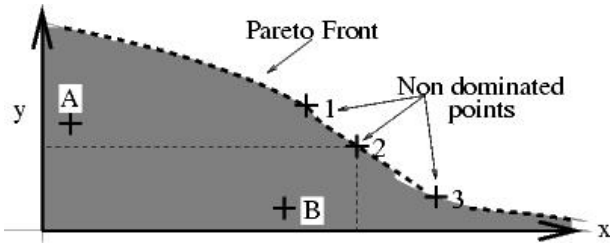


Figure 2.5: Two-dimensional example of Pareto optimality and the Pareto front, where the goal is to maximise along both the x and y axes.

Pareto-Optimal Set and Pareto-Optimal Front

The goal of EMOs is to identify a set of solutions which are non-dominated with respect to each other in the search space. Therefore, the set of solutions cannot be improved in any objective without causing a degradation in at least one other objective. In other terms, these solutions are optimal in the wider sense that no other solutions in the search space are superior to them when all objectives are considered.

The set of non-dominated solutions within the entire search space constitutes the so-called *Pareto-optimal set* or *Pareto-optimal front*. Ideally, one would want to find the *Pareto front*, i.e., the set of all non-dominated solutions in the search space. However, this is often unrealistic, as the size of the *Pareto front* is often limited only by the precision of the problem representation. If x and y in Figure 2.5 are real-valued, for example, and the *Pareto front* is a continuous curve, then it contains an infinite number of points, making a complete enumeration impossible.

Non-dominated Sorting Genetic Algorithm Version II (NSGA-II)

Over the past decade, a number of EMO algorithms that have the ability to find multiple Pareto-optimal solutions in one single run, have been

suggested. NSGA [71] is a popular non-domination based genetic algorithm for multi-objective optimization. Despite being a very effective algorithm, it has been mainly criticized for its high computational complexity, lack of elitism and need for specifying the optimal parameter value for the sharing parameter [21]. In order to address these issues, Deb, et al. [21] developed a fast elitist multi-objective optimization algorithm called Non-dominated Sorting Genetic Algorithm Version II (NSGA-II) which is a much improved version of NSGA. In the following part, NSGA-II is described in detail.

The evolutionary process shown in Algorithm 1 initially starts from generating a random population P_0 of size n which is sorted based on non-domination level. Each individual in the population is assigned a fitness equal to its non-domination rank (1 is the best level). Subsequently a child population Q_0 is created by performing binary tournament selection, simulated binary crossover and polynomial mutation. A combined population R_t is then formed from the parent P_t and offspring Q_t populations which is ranked according to the non-domination relation, and a set of non-dominated fronts F are obtained. The new population P_{t+1} is generated by adding the individuals from the first front and the followings until it exceeds the population size n. Finally, a new child population Q_{t+1} is created by applying selection, crossover and mutation to the parent population P_{t+1}. The preceding procedure is repeated until the maximum number of generations are reached.

In the procedure of NSGA-II, a new parameter value called *crowding distance* is assigned to all the individuals in the population once the non-dominated sort is complete. The crowding distance is a measure of how close an individual is to its neighbours. Large average crowding distance will result in better diversity in the population. As shown in Algorithm 1, individuals are selected from the population by using binary tournament selector based on non-domination rank and crowding distance.

Algorithm 1 The NSGA-II procedure

Generate an initial population P_0 of size n
Rank and sort P_0 based on non-domination level
Apply selection, crossover and mutation to create a child population Q_0 of size n
set t = 0, T = number of generations
while $t < T$ **do**
  R_t = $P_t \cup Q_t$
  Partition R_t into fronts F_1, F_2, ...
  Set $P_{t+1} = \emptyset$, i = 1
  while $\mid P_{t+1} \mid \leq n$ **do**
    Calculate crowding distance in F_i
    if $\mid F_i \mid + \mid P_{t+1} \mid \leq n$ **then**
      P_{t+1} = $P_{t+1} \cup F_i$
    else
      Sort elements of F_i by crowding distance in decreasing order
      P_{t+1} = $P_{t+1} \cup$ the first (n - $\mid P_{t+1} \mid$) elements of F_i
    end if
    i = i + 1
  end while
  Calculate crowded comparison operator $\forall\, i \in P_{t+1}$
  Create a new population Q_{t+1} of size n by applying crossover and mutation to parents selected via binary tournaments on P_{t+1}
  t = t + 1;
end while

Improved Strength Pareto Evolutionary Algorithm (SPEA2)

As one of the most important EMOs that use elitism approach, Zitzler, et al. [106] proposed the modified Strength Pareto Evolutionary Algorithm namely SPEA2 as an improved version of SPEA [105] in 2001. A brief summary of the SEPA2 algorithm is given in Algorithm 2.

The evolutionary process of SPEA2 starts with an initial population P_0 and an empty archive (external set) P'_0. After fitness values are assigned to both archive and population members, all non-dominated individuals in the population and archive are copied to the updated archive P'_{t+1}. If the size of the updated archive P'_{t+1} exceeds the predefined size n', further archive members are removed by means of the truncation operator. The next step represents the mating selection phase where individuals from the union of population and archive are selected by means of binary tournaments. Finally, after crossover and mutation the old population P_t is replaced by the resulting offspring population P_{t+1}. The preceding steps are performed per generation until the maximum number of generations is reached.

In contrast with SPEA, SPEA2 uses a fine-grained fitness assignment strategy. Each individual is assigned a raw fitness calculated based on the strength value of solutions who dominate it. Additionally, it incorporates density information to discriminate between individuals having identical fitness value. After fitness evaluation, all non-dominated individuals from current population and external population are passed into the next generation. If the number of these individuals is less than population size then the next population is filled with dominated individuals from current and external populations.

The size of $archive$ that consists largely of the Pareto front of non-dominated individuals discovered so far is fixed. Whenever the number of non-dominated individuals is less than the predefined archive size, the archive is filled up by dominated individuals. For diversity preservation, SPEA2 utilizes a truncation procedure to replace the clustering technique

Algorithm 2 The SPEA2 procedure

Generate an initial population P_0 of size n, and create an empty archive (external set) P'_0 of size n'
set t = 0, T = number of generations
while $t < T$ **do**
  Calculate the fitness values of individuals in P_t and P'_t
  Copy all non-dominated individuals in P_t and P'_t to P'_{t+1}
  if $\mid P'_{t+1} \mid > n'$ **then**
    Reduce P'_{t+1} by means of the truncation operator
  else
    Fill P'_{t+1} with dominated individuals in P_t and P'_t
  end if
  Perform binary tournament selection with the replacement on P'_{t+1} in order to fill the mating pool
  Apply crossover and mutation to the mating pool and create a new population P_{t+1}
  $t = t + 1$;
end while
Return a non-dominated set represented by the non-dominated individuals in P'_{t+1}

which SPEA uses when the non-dominated front exceeds the archive limit.

2.1.4 Genetic Programming

Evolutionary computation (EC) involves a family of approaches called evolutionary algorithms (EAs) which mimic the biological evolutionary process to resolve optimization problems. This kind of evolutionary process, as illustrated in Figure 2.6, is population-based and involves both competitive and cooperative mechanisms during the evolution of the population. Genetic programming (GP) as a commonly used EA was developed and popularized by John Koza [48] over 20 years ago. In GP, a candidate solution to an optimization problem is represented by a computer program (which normally is a tree structure). GP has been successfully applied to a wide range of problem domains. For example, GP has been used to evolve solutions to problems which has not previously been manually solved by humans [48]. Due to its good global search capability, GP is employed to solve the problem in this work.

In GP, the term $population$ refers to a collection of candidate solutions called $individuals$ to an optimization problem. The term $chromosome$ typically refers to one candidate solution to a problem, which can be represented as a tree that is the most commonly used representation. The major variations of GP include the terminal and function sets. The terminal set consists of the variables and constants of the programs to be constructed. The function set consists of the functions of the programs. In the example of symbolic regression, the terminal set would contain two variables, x and y, and the function set would be composed of different mathematical functions, such as addition, subtraction and division. In addition, GP requires a fitness function to evaluate each individual in current population. The fitness function measures the ability of the encoded individual to solve the problem.

Conventional GP derives new solutions from existing solutions using

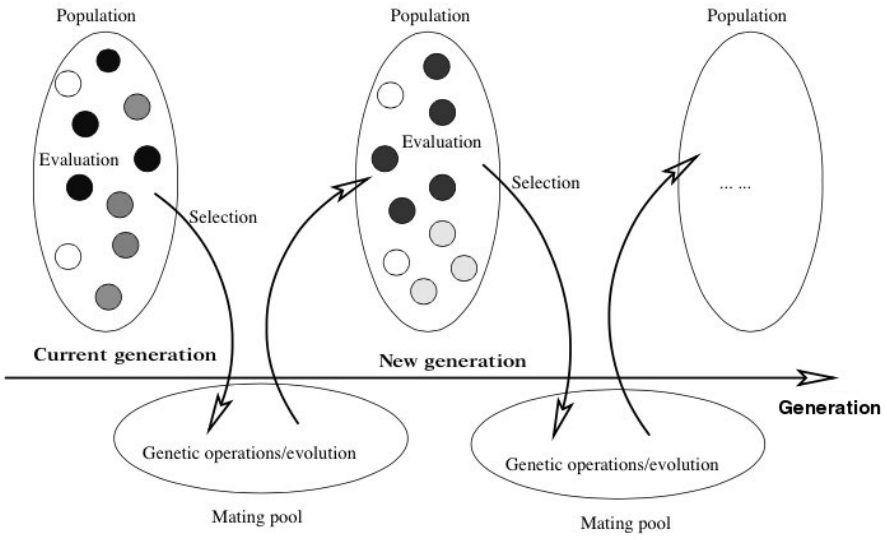


Figure 2.6: The generic evolutionary process for an evolutionary algorithm (EA)

three types of genetic operators which are explained as follows.

- $Selection$. A number of individuals in the population are selected to breed a new generation. Individuals selection is a fitness-based process, where fitter individuals are typically more likely to be selected for reproduction.

- $Crossover$. The crossover operator takes two parents and replaces a randomly chosen part of one parent with another randomly chosen part of the other in order to produce two new offsprings (see Figure 2.7).

- $Mutation$. The mutation operator takes one parent and replaces a randomly selected part of that parent with a randomly generated sequence of code (see Figure 2.7).

A typical GP framework for solving an optimization problem is depicted in Figure 2.8. At the start of an evolutionary run, a population is

initialized to be filled with a group of randomly generated candidate solutions to the problem. The solutions are then evaluated using a fitness function to determine their relative ability to solve the problem and each candidate solution is assigned a respective fitness value. Following that, a selection mechanism is applied to the population, and genetic operations (i.e., crossover and mutation) are performed on the selected solution to generate new candidate solutions for the new population. Therefore, the new population consists of only relatively fit solutions from the initial population and their derivatives. Accordingly, it is expected that the average fitness of the population will have increased. This generational process is repeated over and over again until the number of generations exceeds the specified limit, or any other stopping criteria are satisfied (e.g., an optimal solution is found).

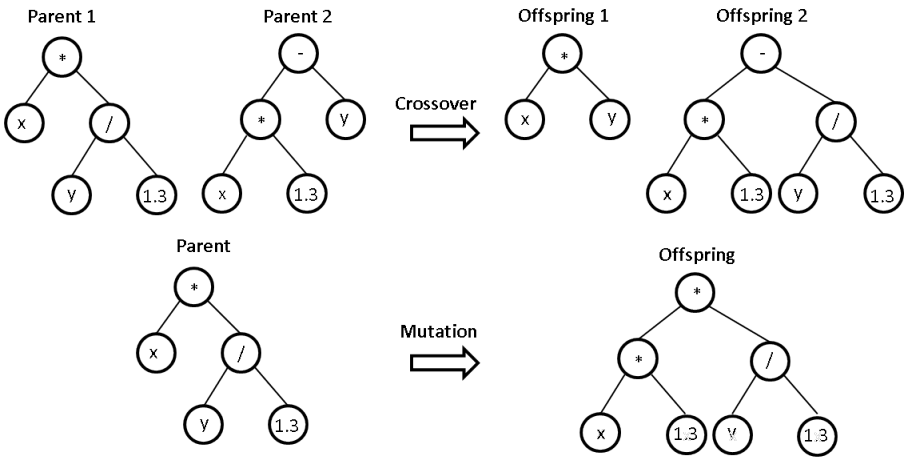


Figure 2.7: An example of crossover and mutation for the symbolic regression problem

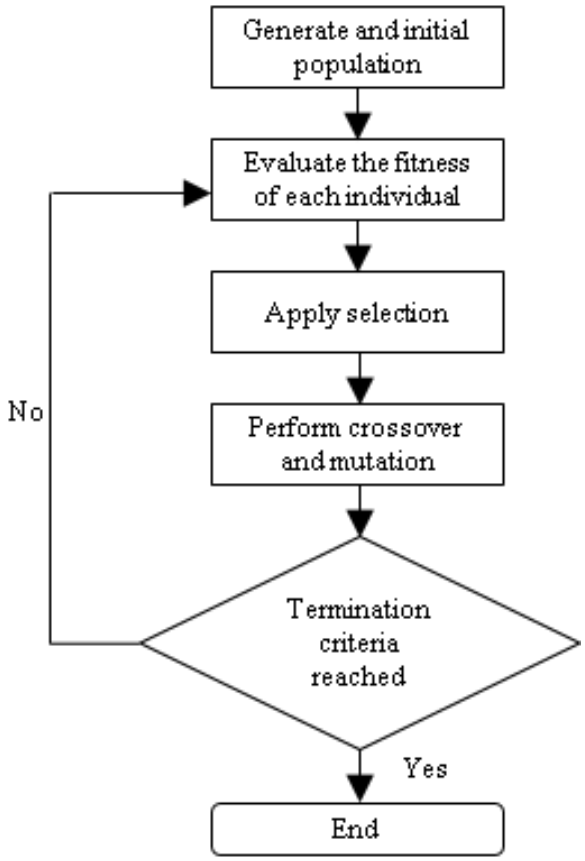


Figure 2.8: The typical procedure used by GP to solve an optimization problem

2.2 An Overview of QoS-Aware Web Service Composition

The procedure for web service composition basically comprises three phases. As illustrated in Figure 2.9, a critical step of web service composition is to select proper web services for composition in order to satisfy users' requests. However, in the presence of numerous web services with equivalent functionality, web services are described and discriminated in terms of both functional capabilities and non-functional (i.e., QoS) properties (e.g., response time, execution cost, availability, etc). Nowadays, web service composition is therefore usually combined with end-to-end QoS requirements, which is *QoS-aware web service composition*. The goal of QoS-aware web service composition is to discover the best composition of web services that meet these end-to-end QoS requirements, while fulfilling the functional requirements. As an example, a user may require to minimize the response time meanwhile satisfying certain constraints in terms of execution cost and reliability, while another user may give more importance to execution cost than to response time.

A typical scenario of web service composition is a "Travel Planner" system depicted in Figure 2.10 that is composed of three tasks for flight booking, hotel reservation and car rental, and concrete services need to be selected for each abstract task. As a result, web services are combined with some workflow patterns (e.g., sequence, parallel, choice, etc), and the eventual composite service can be constrained by some end-to-end QoS requirements. In most of the existing approaches, solving the QoS-aware service composition problem starts with a predefined workflow for a composite web service which contains a set of abstract tasks, and for each abstract task there are a number of available concrete services which provide identical functionality but present different QoS properties. In general, two different optimization strategies could be applied to solve this service composition problem. One of them is the so-called *local op-*

timization where an optimal concrete service is selected for each abstract task independently and the performance of each task is assured. The advantage of this approach is its efficiency as the time complexity of the approach is linear with respect to the number of abstract tasks and concrete services. However, local optimization cannot guarantee the global QoS constraints posed by the user (i.e., minimized total response time and execution cost). Another optimization strategy, called *global optimization*, considers QoS constraints and user's preferences globally which aims to obtain the optimal overall QoS of a service composition. However, this strategy introduces a tradeoff between global optimization and increased complexity. Most of the existing research separates the process of generating a service composition from the process of selecting optimal concrete web services for the given service composition. However, this separation restricts the space of finding optimal service composition solutions. In our study, the generation of a service composition is combined together with concrete service selection.

2.3 Related Work

QoS-aware web service composition introduces a global optimization problem with multiple constraints, and it has received much interest in the past few years. Finding an optimal combination of atomic web services among a large number of possible solutions takes significant computation efforts. A variety of approaches have been put forward to solve the QoS-aware service composition problem. These approaches can be grouped into two categories: approaches based on evolutionary algorithms such as genetic programming (GP), particle swarm optimization (PSO) and ant colony optimization (ACO) and approaches based on non-evolutionary algorithms such as integer programming (IP), graph theory. Each of these approaches is shown in Figure 2.11 and described briefly in the following sections.

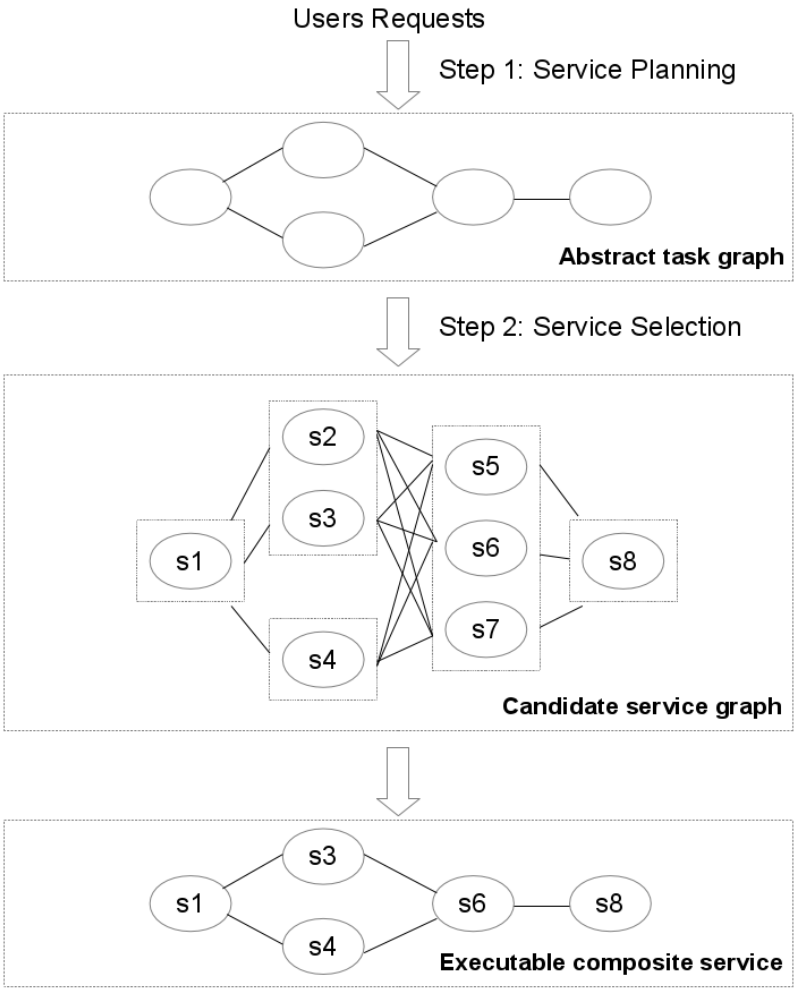


Figure 2.9: The prodecure for QoS-aware web service composition

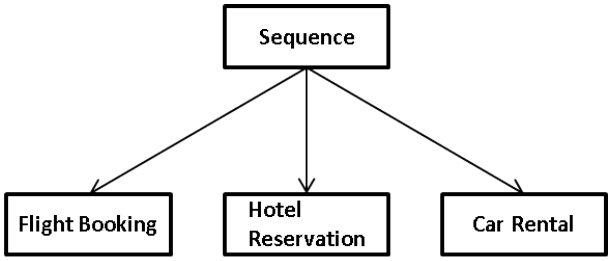


Figure 2.10: An example of a "Travel Planner" system

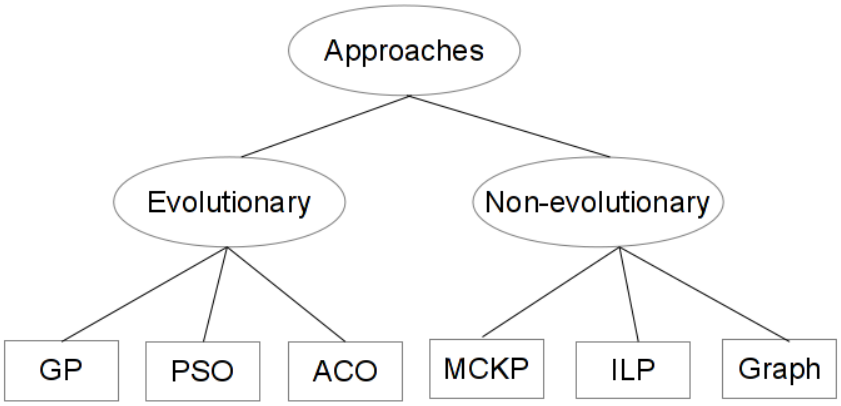


Figure 2.11: Two categories of approches to QoS-aware web service composition problem

2.3.1 Conventional (Non-Evolutionary) Approaches

Over the past decade, there has been a lot of research efforts addressing the end-to-end QoS constraints in web service composition. In [50, 92], the problem of QoS-aware web service composition is regarded as a multiple-choice knapsack problem (MCKP) in which the service composition represents the knapsack and each candidate service represents one item that can be put into the knapsack. The goal of the algorithm is to identify the optimal selection of web services which maximises the value while keeping the constraint on the limited weight capability of the knapsack that is represented by the QoS constraints. However, the time complexity of the algorithms is exponential which makes the MCKP model not suitable for large scale service composition problems. The service composition problem can also be modeled as a graph based on graph theory. In [53], the node in the graph denotes a web service and the edge denotes the interaction between services, whereas in [55], the authors use the node to represent a service orchestration, the edge to represent a web service and each path in the graph to represent a complete service composition. The problem is then transformed into finding the order of nodes in the graph

to generate an executable workflow to satisfy users' requests. Similarly, the graph model is confronted with the same exponential time complexity problem as the MCKP model.

Since local QoS optimization cannot guarantee the global optimization of QoS, some researchers have proposed global QoS optimization approaches that use global planning algorithm and integer linear programming (ILP) to solve the QoS-aware web service composition problem.The authors of [25] put forward a global planning approach, and quality constraints and preferences are assigned to a composite service rather than to the individual tasks involved in the composite service. Service selection is then formulated as an optimization problem and a linear programming method is applied to compute the optimal service execution plans for the service composition. ILP is adopted in [25, 26] to optimally select concrete services in which the objective function is defined as a linear composition of multiple QoS constraints. However, the shortcoming of the ILP technique is that the objective functions and constraints are required to be linear which limits the practicality of the approach. Moreover, linear increase in the size of the service composition problem leads to exponential growth of the algorithm's computational time. If non-linear integer programming is adopted, scalability is still a problem [10]. To overcome these disadvantages, evolutionary approaches have been applied to find the (near)-optimal service composition.

2.3.2 Single-Objective Evolutionary Approaches

Recently evolutionary computation (EC) methods have been more active to overcome the drawbacks of traditional optimization methods. A genetic algorithm (GA) based approach, as a powerful tool to solve combinatorial optimization problems, is proposed in [99] for QoS-aware web service composition. Compared with ILP, GA is able to scale well with respect to the number of web services. The authors of [99] adopt a one-dimensional

chromosome-encoded method in which each gene of the chromosome represents a candidate service. As a consequence, the length of the chromosome increases as the number of tasks and candidate services increases. A revised encoding method is put forward in [11], where each gene of the chromosome represents an abstract task of a composite service, and its value represents a candidate service. However, this encoding schema cannot reflect the relationships between component services in a composite service efficiently. Therefore, a tree-based GP approach is developed in [95] which makes it easier to understand and interpret the relationships between component services. In addition, an adaptive strategy is applied to the search parameters of GP in order to prevent the premature convergence of GP. Yang, et al. [88] use the combination of ACO and GA to tackle the service composition problem. The problem is solved by ACO, however, there are some important parameters in ACO that have great effect on the algorithm. Therefore, they use GA to set the key parameters of ACO in order to obtain a great efficiency. Chen, et al. [58] solve the service composition problem by using DPSO (Discrete Particle Swarm Optimization) which is a variant of standard PSO. In this method they regard each particle as a solution and each particle has a position and velocity. The positions of particles are updated based on their last best position and the best position that has been seen so far. These EC-based methods simply assume that multiple quality criteria, no matter whether they are competing or not, can be combined into a single criterion to be handled by the weighted sum approach. In practice, when users are not certain about the importance of each quality dimension, the weighted sum approach is very difficult to apply because weights are not well known *a priori*. Furthermore, the weighted sum approach largely depends on the formulation (i.e., the weighted formula of the objectives), which has to be readjusted and computed again when the scenario changes.

Although various approaches have been presented to solve the QoS-aware web service composition problem, there is limited work in the lit-

erature for data-intensive service composition [10, 83, 84, 101]. The authors of [10, 101] consider the data intensity of service compositions, but they overlook the communication cost of mass data transfer and its effects on the performance of business processes with different structures. An ant colony based method is proposed in [83, 84] to find the cost minimized data-intensive service composition by considering the access cost and communication cost of mass data transfer. However, the approach focuses on minimizing the cost only without reflecting other important quality dimensions (e.g., availability and reliability).

2.3.3 Multi-Objective Evolutionary Approaches

Despite the multi-objective character of the QoS-aware web service composition problem, EMOs are less used than single-objective algorithms. As discussed before, single-objective optimization approaches have some inherent limitations in solving QoS-aware web service composition. Therefore, several multi-objective heuristics have been proposed. Taboada et al. [73] employ EMO algorithms to provide a set of optimal solutions with different levels of trade-offs for the QoS-aware service composition problem. In [16, 52, 89], the authors resolve the service composition problem using a multi-objective genetic algorithm called NSGA-II [21], without assigning any weight to any quality criterion, and a solution is encoded in the form of an integer vector. The experimental results reveal that NSGA-II is a reasonably good fit in solving the QoS-aware service composition problem with satisfied convergence and distribution properties. Similarly, Li, et al. [51] adopt another EMO algorithm called SPEA2 to solve the service composition problem in which three performance criteria (i.e., response time, execution cost and availability) are considered. The same type of genome encoding, namely integer vector based, is used. Since the experiments are conducted based on different datasets and different number of objectives, it is impossible for us to compare the performance of the two EMO

algorithms for solving QoS-aware web service composition. Cremene, et al. [18] apply both EMO algorithms (i.e., NSGA-II and SPEA2) to the service composition problem, however, mere two quality criteria (i.e., time and cost) are considered in the simulation experiments. As a result, it is not convincing enough to demonstrate the performance differentiation of the two algorithms. The experiments presented in the preceding literature [16, 18, 51, 52, 89] use a small number of abstract tasks and some of them also use a small number of candidate services. Therefore, it is not obvious how scalable these approaches are. Claro et al. [17, 91] also discuss the advantages of two popular EMO algorithms, NSGA-II and SPEA2 in finding optimal service compositions. In contrast to single-objective optimization approaches, these EMO based approaches do not require user to prioritise, scale, or weight objectives in advance, which is more realistic and reliable. Moreover, reformulating the solutions is not required if there is a change (e.g., the change of the user's preferences). However, existing methods consider two or three quality objectives only due to the limitations of current EMO algorithms in higher dimensional objective space.

2.3.4 Decentralized Execution of Composite Web Services

In recent years, many researches have been emerging to address the problem of decentralized execution of composite web services. A top down approach for integrated process modelling and distributed process execution is proposed in [68] without considering data dependencies. In the Mentor project [86], a workflow is partitioned for decentralized execution based on rules and state charts into a set of sub-workflows which are then enacted by a number of distributed workflow engines. Khalaf et al. [44, 45] present a method for partitioning a business process so that each partition can be enacted by a different participant and a corresponding BPEL process along with necessary deployment information is created for each participant. Yildiz et al. [90] propose a method for deriving distributed

processes of a centralized process with respect to their information flow policies. Nanda et al. [61] review previous approaches and introduced two novel heuristic algorithms which are Merge-by-Define-Use (MDU) and Pooling-and-Greedy-Merge (PGM), respectively. They support automatic process partitioning based on an analysis of a program dependence graph generated for the process. However, all of the above approaches do not consider the communication overhead induced by data-flows in the context of data-intensive processes, which in turn has a significant impact on the degree of customer satisfaction. In [75], a dynamic workflow model fragmentation algorithm is proposed to partition a centralized process model into fragments step by step while the process is executed. Although data-intensive tasks are considered in the work, experiments are not conducted to assess the effectiveness and efficiency of the approach.

2.4 Summary

In summary, very few efforts about QoS-aware data-intensive web service composition have been made. The majority of the existing research overlooks the data intensity involved in data-intensive service compositions and the most recent research studies examine the communication cost of mass data transfer only without investigating other important quality dimensions. Meanwhile, as the number of quality objectives increases, the past single-objective and multi-objective optimization approaches are not able to solve the service composition problem with such a high-dimensional objective space. Therefore, there exists a demand for finding an efficient approach which has the ability to cope with the preceding issues. Furthermore, although the decentralized execution of composite web services has been widely studied in the past few years, there is still lack of research on finding an efficient execution plan for data-intensive service compositions in a reasonable time frame.

Chapter 3

A Hybrid GP-Tabu Approach to QoS-Aware Data-Intensive Web Service Composition

As discussed earlier, few efforts have been made so far to investigate the problem of QoS-aware data-intensive web service composition. To the best of our knowledge, the use of genetic programming in data-intensive service composition was not examined in the past research. This chapter is devoted to presenting a GP-based solution to our first research problem of the thesis, i.e., the problem of QoS-aware data-intensive service composition.

3.1 The Problem

With the increasing presence and adoption of web services on the World Wide Web, the composition strategy of a variety of services provides maximal flexibility when designing and implementing data-intensive applications. On the other hand, with the presence of numerous web services with equivalent functionality, web services are described and discriminated in terms of both functional capabilities and non-functional (i.e., QoS) prop-

erties (e.g., response time, execution cost, availability). As a considerable amount of data needs to be exchanged between the components in a composite data-intensive service, the movement of mass data influences the performance of the whole composite web service. Therefore, it becomes inevitable to take into account non-functional properties during the dynamic process of data-intensive service composition, especially attention should be drawn to the access cost and transfer cost of the data between collaborating web services. Therefore, the goal of *data-intensive service composition* is to find the best composition of data-intensive services that meet these non-functional requirements, while fulfilling the functional requirements. As an example, a user may require to minimize the response time meanwhile satisfying certain constraints in terms of execution cost and reliability, while another user may give more importance to execution cost than to response time.

In this chapter, we present a hybrid GP-Tabu approach to the problem of QoS-aware data-intensive service composition. Tabu search (TS) [30] as a meta-heuristic local search is integrated into the evolutionary process of genetic programming (GP) [48] in order to overcome the downsides of GP such as its prematurity and proneness to trap in local optima. The main contributions of this chapter are two-fold. First, a QoS-aware mathematical model is developed to take into account the effect of mass data transfer. Second, a hybrid approach that combines the use of GP and TS is proposed to address the QoS-aware data-intensive service composition problem. The experimental results demonstrate the effectiveness and efficiency of the approach, especially it offers better performance than traditional optimization techniques.

The rest of this chapter is organized as follows. Section 3.2 discusses the time and cost aware model used in the context of data-intensive service composition. In Section 3.3, the details of the proposed approach are explained, and Section 3.4 reports on the experimental results. Finally, the conclusions are drawn in Section 3.5.

3.2 A Time and Cost Aware Model for Data Intensive Services

Since a data-intensive service s_i is provided by a service provider and deployed on a server, a data-intensive service composition SC that consists of a set of m service servers and certain composition patterns can be denoted by SC = {S_1, S_2, ..., S_m}. Each atomic data-intensive service s_i is associated with a QoS vector Q_i = [l_i, c_i, r_i, a_i] where l_i, c_i, r_i, a_i represents the latency, execution cost, accuracy and availability of the service. Assume that a data-intensive service s_i requires a set of data denoted by D_i to form part of its inputs. The latency l_i for the service s_i is made up of three parts: the queue time q_i, the processing time p_i and the transfer time t_i, as shown in Equation 3.1.

$$l_i = q_i + p_i(D_i) + t_i(D_i, S_j, S_i) \tag{3.1}$$

where q_i is the time spent in waiting in the queue for the data D_i to be processed by the server, $p_i(D_i)$ is the actual time used to process the set of data D_i, and $t_i(D_i, S_j, S_i)$ is the data transfer time for transferring the data set D_i from the server hosting the service s_j to the server hosting the service s_i.

To be detailed, the queue time q_i depends on server load, i.e., the current request queue length on the server, while the processing time p_i and the transfer time t_i can be calculated using Equation 3.2 and 3.3.

$$p_i(D_i) = size(D_i)/pr(S_i) \tag{3.2}$$

$$t_i(D_i, S_j, S_i) = size(D_i)/bw(S_j, S_i) + size(D_i)/ws(S_i) \tag{3.3}$$

where $size(D_i)$ is the size of the data set D_i, $pr(S_i)$ denotes the processing rate of the server S_i, $bw(S_j, S_i)$ is the network bandwidth between the server S_j and the server S_i, and $ws(S_i)$ is the disk write speed of the server S_i. To sum up, as described in the above equations, the processing time

depends on the processing capacity of the server, and the data transfer time is determined by the network bandwidth and the amount of data to be transferred between two service servers. As each server has many requests at the same time and it serves only one request at a time, the current service request needs to wait until all requests prior to it in the queue have completed.

In addition to time, the costs generated by data-intensive services as well as the movement of mass data have a significant impact on the total cost of a service composition. Consider a data-intensive service s_i, similarly its execution cost c_i consists of three parts: the data access cost ac_i, the data transfer cost tc_i, and the service related cost sc_i. As can be seen from Equation 3.4, the data access cost ac_i is the price to be paid for writing the data D_i to the server that hosts the service s_i and reading the data in order to invoke the service. The data transfer cost tc_i is proportional to the size of the data set D_i, which depends on the available network bandwidth between two service servers. The service related cost sc_i expresses the cost to provision the service s_i as well as the cost to process the service request including data processing.

$$c_i = ac_i(D_i) + tc_i(D_i, S_j, S_i) + sc_i(D_i) \tag{3.4}$$

$$ac_i(D_i) = size(D_i) * wcost(S_i) + size(D_i) * rcost(S_i) \tag{3.5}$$

$$tc_i(D_i, S_j, S_i) = size(D_i) * tcost(S_j, S_i) \tag{3.6}$$

$$sc_i(D_i) = pcost(S_i) + size(D_i) * dcost(S_i) \tag{3.7}$$

where $size(D_i)$ denotes the size of the data set D_i, $wcost(S_i)$ is the cost of writing per unit of data to the server S_i, $rcost(S_i)$ is the cost of reading per unit of data from the disk on the server S_i, $tcost(S_j, S_i)$ is the transfer cost from the service server S_i to the service server S_j for per unit of data, $pcost(S_i)$ is the price charged to use the service s_i which is usually specified by service provider, and $dcost(S_i)$ is used to represent the expenditure for processing each unit of data on the server S_i.

For each data-intensive service, the other two QoS attributes (i.e., accuracy and availability) are supposed to have fixed values which can be collected from service providers. Therefore, the QoS-aware data-intensive service composition can be regarded as an optimization problem. Clearly the goal of the optimization problem is to minimize the latency and execution cost that have been defined in Equations 3.1 and 3.4, meanwhile achieving the maximum possible accuracy and availability for a composite web service.

3.3 The Hybrid GP-Tabu Approach

Artificial intelligence techniques have been widely used to solve many optimization problems. In recent years, GP has become increasingly popular as an alternative to more classical techniques in science and engineering disciplines. As another powerful optimization procedure, TS is capable of escaping local optimum trap by employing a flexible memory system, and it has been successfully applied to a diverse range of combinatorial optimization problems. These methods seem to be promising and are still evolving. Next, the TS method is briefly reviewed before the proposed hybrid approach is presented.

3.3.1 An Overview of Tabu Search

TS [30] is a meta-heuristic that guides a local heuristic search procedure to explore a problem's solution space with the goal of avoiding local optimum and ultimately finding the desired solution. The basic principle of TS is to avoid cycling back to previously visited solutions and allow non-improving moves whenever a local optimum is encountered. This is achieved by using a short-term memory that records the recent history of the search to prevent investigating the solution space that has been visited before. However, in some situations, TS permits backtracking to previ-

ous solutions which may ultimately lead to better solutions via a different direction. The two main components of TS are the tabu list and the aspiration criteria of the solution associated with the recorded moves.

- *Tabu list*. Certain forbidden moves (trial solutions) are maintained in the list to prevent cycling when moving away from local optimum through non-improving moves. As a result, the search is not allowed to return to a recently visited point in the search space, that is, a recent move is not allowed to be reversed. Usually the tabu list stores a fixed or fairly limited quantity of information. Empirically, the size of the list that provides good results often grows with the size of the problem, and stronger restrictions are generally coupled with smaller list size.

- *Aspiration criteria*. A key issue for tabu list is that it is sometimes so powerful to prohibit attractive moves, even cycling cannot occur, or they may lead to an overall stagnation of the search process. Hence, aspiration criteria are used to allow for exceptions from the tabu list, if such moves lead to promising solutions. The simplest and most commonly used aspiration criterion, found in almost all TS implementations, allows a tabu move when it results in a solution with an objective value better than that of the current best-known solution (since the new solution has obviously not been previously visited).

3.3.2 The Proposed Hybrid Approach

In order to determine a solution to the QoS-aware data-intensive service composition problem, we propose a new hybrid approach where the evolutionary process of GP works along with the local TS search procedure. To be specific, the proposed hybrid approach adopts the neighbour solutions found by TS to generate part of a new population in the global search process of GP. The major steps of our approach are described in Algorithm 3. The approach starts from a randomly initialized population

after setting up the necessary variables for GP and Tabu. The individuals in the current population are then evaluated using the specific fitness function. Crossover and mutation are performed on the selected individuals to produce next generation. For every t generations, the m best individuals (i.e., service compositions) in the population are selected as the initial solutions of the TS procedure. As a result, n neighbour solutions are generated by mutating a random node in the tree representation of a candidate service composition solution, and the n worst individuals in the current population are replaced. The above process is repeated until the maximum number of iterations is reached.

Algorithm 3 GP-Tabu for QoS-aware data-intensive service composition

Require: available inputs, required outputs, QoS constraints and a service repository

Ensure: a service composition that meets both functional and non-functional requirements

1: Initialize the parameters of GA and TS, and set g=1
2: Generate an initial population P randomly
3: Evaluate each individual i in P using the fitness function
4: **while** $g < g_{max}$ **do**
5: Select two parents from the population P. Perform crossover with rate P_c, and perform mutation with rate P_m to generate a new population P'
6: **if** $g \bmod k = 0$ **then**
7: Choose the m best individuals from the current population, and apply the TS algorithm to generate n neighbours to substitute the worst n individuals in the new population P'
8: **end if**
9: Evaluate each individual i' in P' using the fitness function
10: Set g=g+1
11: **end while**
12: **return** the individual with the best fitness

In our approach, the fitness function introduced to measure the performance of each individual i in the gth generation of the evolution process is defined as follows.

$$f_i = \frac{(w_1 R_i + w_2 A_i) * (w_5 I_i + w_6 O_i)}{w_3 L_i + w_4 C_i} \tag{3.8}$$

where R_i, A_i, L_i and C_i denote the aggregated accuracy, availability, latency and execution cost of a composite data-intensive service, each of which can be calculated using the formulae described in Table 2.1 and the equations proposed in Section 3.2. For example, the aggregated latency for a sequence workflow structure is $L = \sum_{i=1}^{j} l_i$. In the function, w_1, w_2, w_3, w_4, w_5 and w_6 are real and positive weights. A larger weight means that that particular QoS attribute is considered more important than others from the point of view of users. Note that the fitness function can be easily adapted to user's requirements, i.e., adding or removing QoS attributes without affecting the performance of our approach. I_i and O_i that indicate the degree to which a valid solution has been found are presented in Equation 3.9.

$$I_i = \frac{|input_r|}{|input_r \bigcup input_a|} \qquad O_i = \frac{|output_r \bigcap output_a|}{|output_r|} \tag{3.9}$$

where $input_r$ is the list of inputs available for a composite service solution, $input_a$ is the list of inputs required by the solution, $output_r$ is the list of outputs desired by a composition task, $output_a$ is the list of outputs that are actually produced by the solution, and $|.|$ represents the size of the list.

To guarantee incommensurable QoS attributes have fair impact on the calculation of fitness, the value of each QoS attribute is to be normalized in the interval [0,1]. All the weights utilized in the fitness function also falls within the range [0, 1]. As illustrated in Equation 3.8, QoS-aware data-intensive service composition is converted to a maximization problem, i.e., greater fitness denotes more satisfying solution.

3.4 Experimental Studies

3.4.1 Test Cases

For the purpose of evaluation, we carry out a set of experiments using the test cases provided by the public benchmark datasets, WSC2008 [7] and WSC2009 [47]. Each dataset consists of a great number of web services associated with randomly generated inputs and outputs. However, QoS attributes are not included in neither datasets. Therefore, the QoS values of web services are generated based on the data collected in another public dataset called QWS [2]. Each test case is made up of available inputs, required outputs, and a service repository. The complexity of the test cases is diverse in terms of the number of atomic web services and the number of workflow structures involved.

3.4.2 Parameter Configurations

The experiments are conducted with the population size of 200 for maximum 100 generations (i.e., g_{max}=100). The crossover probability P_c=0.9, the mutation probability P_m=0.2, and the size of the tabu list is 7. In our experiments, the top 10% individuals in the current population (i.e., m=20) will be selected for applying TS for every 10 generations (i.e., t=10), so the 20 worst (i.e., n=20) individuals will be replaced with the new neighbour solutions. Assume that availability and execution cost are considered more important than accuracy and latency. The weights defined in the fitness function are $w_1 = 0.2$, $w_2 = 0.3$, $w_3 = 0.2$, $w_4 = 0.3$, $w_5 = 0.5$ and $w_6 = 0.5$ which can give better performance indicated by a large number of empirical trials. Since our approach is non-deterministic, 30 independent runs are performed for each test case.

To simulate the time and cost for data access and transfer in our experiments, the amount of input data for an atomic data-intensive service is randomly generated in the interval (0, 30]MB, and the amount of out-

put data is determined by multiplying by a random factor of 0∼10. For the sake of simplicity, the queue time required by a service server is between 0∼10s, all server's data write speed (i.e., $ws(S_i)$) and process rate (i.e., $pr(S_i)$) are both 10MB/s, the transfer rate between two service servers (i.e., $bw(S_j, S_i)$) is 3MB/s, and all other costs such as data access cost and transfer cost for per unit of data are all 1.

3.4.3 Experimental Results and Analysis

The experimental results for the test cases are presented in Table 3.1. Each row in the table shows the fitness of the solutions found by GP and GP-Tabu for each test case. To demonstrate the superiority of our approach over the simple GP method, a significance test (z-test) is conducted to compare the solutions found by the two approaches. As illustrated in the table, for simple composition tasks (i.e., WSC2008-1, WSC2008-2 and WSC2008-4), both GP and GP-Tabu are able to make the same *optimal* service composition to achieve the high-level task. However, it is observed that for more complicated test cases, the GP-Tabu approach is capable of finding better compositions of services indicated by a significant improvement on the fitness. In summary, our hybrid approach was successful in computing a solution to each of the service composition tasks, and the results showed that it is much more effective for complicated tasks.

Further Analysis

To further study the effectiveness and efficiency of our approach, here we conduct a set of experiments with the same test cases on two traditional optimization methods, i.e., TS and integer linear programming (ILP) [11]. In order to evaluate the quality of the solutions found by different optimization methods, a unity function that adopts the simple additive weighting approach is used as shown in Equation 3.10.

$$U(SC) = w_1 \cdot R + w_2 \cdot A + w_3 \cdot L + w4 \cdot C \tag{3.10}$$

Table 3.1: The results of the test cases for GP and GP-Tabu. (↑ denotes significantly better)

Test case	GP	GP-Tabu
WSC2008-1	0.9167 ± 0.0000	0.9167 ± 0.0000
WSC2008-2	0.9206 ± 0.0000	0.9206 ± 0.0000
WSC2008-3	0.9025 ± 0.0017	$0.9114 \pm 0.0009\uparrow$
WSC2008-4	0.8668 ± 0.0000	0.8668 ± 0.0000
WSC2008-5	0.8126 ± 0.0014	$0.8135 \pm 0.0008\uparrow$
WSC2008-6	0.8461 ± 0.0003	$0.8556 \pm 0.0002\uparrow$
WSC2008-7	0.8988 ± 0.0008	$0.9052 \pm 0.0013\uparrow$
WSC2008-8	0.8825 ± 0.0011	$0.8857 \pm 0.0007\uparrow$
WSC2009-1	0.8276 ± 0.0002	$0.8311 \pm 0.0006\uparrow$
WSC2009-2	0.8844 ± 0.0001	$0.8982 \pm 0.0011\uparrow$
WSC2009-3	0.7846 ± 0.0023	$0.7931 \pm 0.0009\uparrow$
WSC2009-4	0.7024 ± 0.0006	$0.7546 \pm 0.0019\uparrow$
WSC2009-5	0.7290 ± 0.0008	$0.8449 \pm 0.0023\uparrow$

where w_1, w_2, w_3, and w_4 remain the same as described in Section 3.4.2. I and O are not included in the function as they are specified as constraints in all the optimization methods, that is, the solution found must be able to generate the desired outputs given the available inputs.

The simulation results of the experiments are shown in Table 3.2. The last three columns of the table present the fitness of the solutions found by GP-Tabu, ILP and Tabu, respectively. As can be observed from the table, the ILP method cannot find a valid solution for most of the service composition tasks except for tasks WSC2008-1, WSC2008-2, WSC2008-4 and WSC2008-5, and the significance test demonstrates the superiority of the GP-Tabu approach over ILP in this problem domain. In contrast, the TS method is able to find a service composition solution for each task, especially when the tasks (i.e., WSC2008-1, WSC2008-2, WSC2008-3, WSC2008-4, WSC2008-5 and WSC2009-2) are relatively simple. However, in most situations where the service composition request is very complex, our GP-Tabu approach is recommended due to its better performance implied by the significant improvement from the statistical test.

Table 3.2: The results of the test cases for GP-Tabu, ILP and Tabu. (↓ denotes significantly worse, and ↑ denotes significantly better)

Test case	GP-Tabu	ILP	Tabu search
WSC2008-1	0.5946 ± 0.0000	$0.5849 \pm 0.0014\downarrow$	$0.5916 \pm 0.0019\downarrow$
WSC2008-2	0.4997 ± 0.0000	$0.4654 \pm 0.0006\downarrow$	$0.5108 \pm 0.0007\uparrow$
WSC2008-3	0.4588 ± 0.0005	n/a	$0.4836 \pm 0.0008\uparrow$
WSC2008-4	0.4738 ± 0.0000	$0.4531 \pm 0.0013\downarrow$	$0.4876 \pm 0.0006\uparrow$
WSC2008-5	0.4888 ± 0.0007	$0.3626 \pm 0.0027\downarrow$	0.4884 ± 0.0010
WSC2008-6	0.4212 ± 0.0003	n/a	$0.4192 \pm 0.0014\downarrow$
WSC2008-7	0.4177 ± 0.0006	n/a	$0.3791 \pm 0.0006\downarrow$
WSC2008-8	0.5146 ± 0.0009	n/a	$0.5033 \pm 0.0012\downarrow$
WSC2009-1	0.6666 ± 0.0008	n/a	$0.5779 \pm 0.0007\downarrow$
WSC2009-2	0.4447 ± 0.0015	n/a	0.4441 ± 0.0009
WSC2009-3	0.5612 ± 0.0008	n/a	$0.5531 \pm 0.0004\downarrow$
WSC2009-4	0.4623 ± 0.0006	n/a	$0.3145 \pm 0.0011\downarrow$
WSC2009-5	0.4844 ± 0.0019	n/a	$0.3487 \pm 0.0023\downarrow$

3.5 Summary

In this chapter, a novel combination of genetic programming and tabu search for solving the problem of QoS-aware data-intensive service composition has been presented. A time and cost aware mathematical model was developed for describing the effect of the movement of mass data. In the proposed approach, the local search procedure employed by TS was integrated into the global search process of GP, in order to avoid premature convergence and getting stuck in local optima. To verify the effectiveness and efficiency of the proposed hybrid approach, it was successfully applied to two public benchmark datasets, i.e., WSC2008 and WSC2009, each of which consists of a large variety of web services as well as diverse service composition tasks. Compared to the simple GP and two traditional optimization methods, the analysis of the experimental results showed the superiority of our approach in finding more satisfying service compositions.

Chapter 4

F-MOGP: A Novel Many-Objective Evolutionary Approach to QoS-Aware Data-Intensive Web Service Composition

Conventional optimization techniques for solving QoS-aware web service composition have been criticised due to their scalability bottlenecks [72, 96]. To that end, genetic programming based and genetic algorithm based methods [12, 15, 41, 93] from the evolutionary computation field have emerged recently for efficiently exploring a huge search space. However, the majority of these methods avoid the complexities involved in the many-objective QoS-aware web service composition problem and transform many objectives into a single-objective function. Likewise, our previously proposed hybrid approach in Chapter 3 simply aggregates multiple (and often conflicting) quality objectives into a higher scalar function for fitness calculation based on a weight mechanism. This approach is suitable for experienced users who have a clear idea about the importance of each

quality objective. However, for the other users who are unsure about the importance of each QoS dimension, the aggregation methods such as the weighted sum approach are very difficult to apply because utility functions or weights are not well known *a priori*. Therefore, the aim of this chapter is to describe how to address the problem of many-objective QoS-aware data-intensive web service composition which is listed as the second objective of this thesis in Chapter 1.

4.1 The Problem

In the past few years, evolutionary multi-objective optimization (EMO) algorithms such as NSGA-II [19] and SPEA2 [103] have been proposed to overcome the aforementioned shortcomings. Compared with returning only a single best solution per run, EMO algorithms are able to simultaneously optimize multiple and often competing objectives and find a set of Pareto-optimal solutions. However, these EMO algorithms suffer from their ability to handle the optimization problems having more than three objectives. The primary reason is that almost all the solutions in the population become non-dominated as the number of objectives increases. In real life situations, on the other hand, it is not uncommon that at least four quality objectives will be considered for a task of QoS-aware service composition. That said, QoS-aware web service composition is intrinsically a "many-objective" optimization problem. Despite all the previous efforts, an efficient many-objective approach is still highly demanded to address the QoS-aware service composition problem. In addition, today data-intensive services based applications have become the most challenging type of applications in service-oriented paradigm. However, the problem of QoS-aware web service composition has not been widely studied in the context of data-intensive workflows. In particular, traditional methods [84, 83] overlook the communication time delay and the communication cost in mass data transfer. As a consequence, these methods have an

inaccurate measure on the performance of the service compositions found.

In order to tackle the QoS-aware service composition problem with four or more quality objectives in the context of data-intensive workflows, a many-objective evolutionary approach named F-MOGP is proposed in this chapter. Based on a reduced space searching strategy, F-MOGP employs a recently developed many-objective optimization algorithm, NSGA-III [20], to find and return a set of optimal service compositions according to the restrictions on various QoS attributes. The experimental results show its effectiveness and ability to exhibit the trade-offs between different solutions in satisfying different objectives.

The rest of this chapter is organized as follows. Section 4.2 presents the implementation details of the proposed F-MOGP approach. Experimental studies are described in Section 4.3, and finally, conclusions are outlined in Section 4.4.

4.2 The Proposed F-MOGP for Many-Objective QoS-Aware Data-Intensive Service Composition

In this chapter, we propose a filter-based many-objective evolutionary approach, named *F-MOGP*, to find a set of data-intensive service compositions which have the optimal values for each quality objective, and offer the option to assess the trade-offs between different solutions. For example, a user can choose the service compositions with high latency and low cost, or the ones with low latency and high cost.

Amongst the four QoS attributes described in Chapter 2, latency and execution cost are negative quality criteria. In other terms, the grades of these criteria increase as their values decrease. In contrast, availability and accuracy are positive quality criteria for which the grades increase as their values increase. Since different QoS attributes are not commensurable to

some extent, the value of each attribute q for a service composition sc is standardized according to Equation 4.1.

$$q^{normal}(sc) = \begin{cases} \frac{Q_{max} - q(sc)}{Q_{max} - Q_{min}}, & \text{for positive quality criteria} \\ \\ \frac{q(sc) - Q_{min}}{Q_{max} - Q_{min}}, & \text{for negative quality criteria} \end{cases}$$

$$Q_{max} = \underset{\forall sc \in S}{max}\, q(sc), \quad Q_{min} = \underset{\forall sc \in S}{min}\, q(sc) \tag{4.1}$$

where S denotes the set of all service compositions found so far. With normalization, the objectives of all the four QoS-based quality dimensions (i.e., latency, execution cost, availability and accuracy) are to be minimized and we assume that all the objectives are equally important in this work. That said, the goal of many-objective QoS-aware data-intensive service composition is to simultaneously optimize all the objectives and obtain the lowest possible QoS for a service composition sc while satisfying the functional requirements, as shown by the objective vector in Equation 4.2. The aggregated values of each QoS attribute can be calculated using the same mathematical model proposed in Chapter 3.

$$minimize \begin{bmatrix} L^{normal}(sc), & C^{normal}(sc), \\ A^{normal}(sc), & R^{normal}(sc) \end{bmatrix} \tag{4.2}$$

4.2.1 Search Space Reduction

The size of the search space is exponential to the number of web services available, which has become a challenging problem that limits the performance of many existing methods. To enhance the performance of our proposed approach, as an initial step of F-MOGP before applying global optimization, an algorithm is developed to prune the search space to a reasonable size. This search space reduction algorithm originates from an idea that are relevant to the dependencies between web services, which are defined in Definitions 1 and 2.

Definition 1 *Given two data-intensive services s_i and s_j, assume the outputs of s_i are O_i and the inputs of s_j are I_j. If $\forall\ i \in I_j$, $i \in O_i$, service s_j is then said to be fully dependent on service s_i, denoted by $s_j \Rightarrow s_i$.*

Definition 2 *Given two data-intensive services s_i and s_j, assume the outputs of s_i are O_i and the inputs of s_j are I_j. If $\exists\ i \in I_j$, $i \in O_i$, service s_j is then said to be partially dependent on service s_i, denoted by $s_j \rightarrow s_i$.*

In Algorithm 4, I and O denote the available inputs and the desired outputs for a service composition task respectively, and R contains a set of data-intensive services available in this task. Initially the services in R that could take I as inputs are fed into S_{found} and they are removed from R to avoid multiple visits. I is then updated to include the outputs generated by all the services in S_{found}. During the following iteration, if a service in R is fully or partially dependent on any of the services in S_{found}, it will be added to S_{found} and removed from R. Finally, when the members of S_{found} no longer change, R^* which has the same services as S_{found} is returned as the output of the algorithm. By examining the dependencies among web services in the original search space using Definitions 1 and 2, we eventually reach a smaller space which contains the web services only related to a given service composition task.

4.2.2 Many-Objective Optimization Based on NSGA-III

After the reduction of the search space, the second phase of F-MOGP is to apply an optimization algorithm to the minimization problem described before. Due to its effectiveness in high dimensional objective space, NSGA-III [20] is adopted as the basis of our proposed many-objective evolutionary approach. However, instead of the chromosome representation, a tree-based structure is adopted as the representation of a service composition in our approach. Web services are denoted by leaf nodes and workflow patterns (e.g., sequence) are expressed by intermediate nodes. The advan-

Algorithm 4 The algorithm for reduction of the search space

Require: I, O, R

Ensure: R^*

$R^* \leftarrow \emptyset$, $S_{found} \leftarrow \emptyset$
for $each\ s \in R$ **do**
if $s_{input} \subset I$ **then**
$S_{found} = S_{found} \cup s$
end if
end for
while $S_{found}\ is\ changed$ **do**
$R^* \leftarrow R^* \cup S_{found}$
$R \leftarrow R \setminus S_{found}$
for $each\ s \in S_{found}$ **do**
$I = I \cup s_{output}$
end for
for $each\ s \in R$ **do**
for $each\ s^* \in S_{found}$ **do**
if $s \Rightarrow s^*\ or\ (s \rightarrow s^*\ and\ s_{input} \setminus s^*_{output} \subset I)$ **then**
$S_{found} = S_{found} \cup s$
end if
end for
end for
end while
$return\ R^*$

tage of representing a service composition as a tree is that understanding and interpreting the relationships between constituent services in a composite web service become easier and more effective. For example, as illustrated in Figure 4.1, it is clear to see that web services WS_1 and WS_2 are executed sequentially, and WS_3, and WS_4 are executed in parallel. The inputs of WS_1, WS_3 and WS_4 come from a user directly, while WS_2 also relies on the outputs of WS_1. Finally the composition of WS_1, WS_2, WS_3 and WS_4 produces the required outputs based on the given inputs and return them to the user. Based on this tree-like structure, the *crossover* operation is performed on two stochastically selected individuals (i.e., service compositions in our case). If the individuals contain two nodes that represent compatible inputs and outputs, the two nodes along with their subtrees are then swapped between these two individuals. This guarantees that the offspring generated is feasible. The *mutation* operator randomly selects a node of a service composition and it is then replaced with a new generated one that has compatible inputs and outputs.

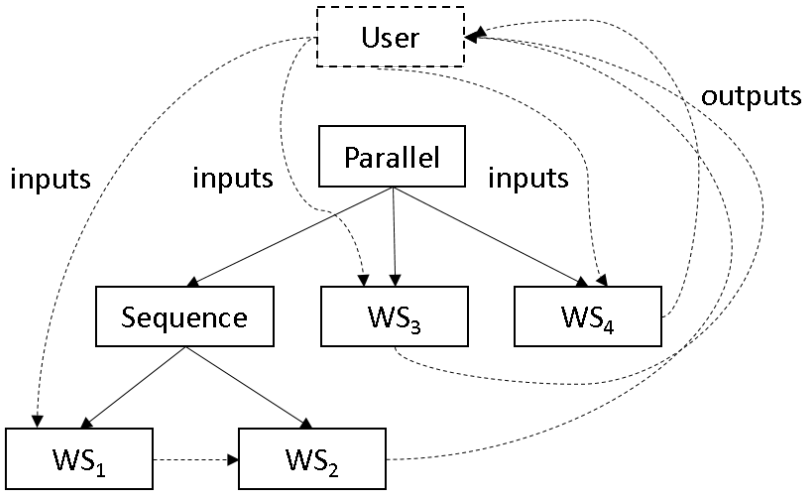


Figure 4.1: A tree-based representation of a service composition.

As shown in Algorithm 5, the optimization procedure starts with multiple reference points and a randomly initialized population with N service compositions. The reference points can be supplied by user based on the preference information. However, we assume that all objectives are equally important and are all to be minimized. Therefore, a set of reference points are generated using Das and Dennis's systematic approach as described in [20] for our optimization problem. At the t^{th} generation, an offspring population Q_t is produced from its parent population P_t using binary tournament selection, crossover and mutation. P_t and Q_t are then combined into a new population R_t of size $2N$. The values of the four quality objectives are calculated and normalized for each service composition in R_t, then these service compositions are sorted into different levels (F_1, F_2 and so on) based on usual domination principle. As the last accepted level, only part of the members in F_l will be selected and added to the population S_t for next generation. To maintain the genetic diversity among individuals, the reference point that has the smallest number of associated service compositions will be used to select members from F_l to S_t. The above process is repeated until the termination criterion is satisfied and a set of Pareto optimal service compositions are eventually found.

4.3 Experimental Studies

4.3.1 Test Cases

To assess the performance of our proposed approach, we carried out a set of experiments using the same datasets as the ones used in Chapter 3. Each test case in the datasets is comprised of available inputs, required outputs, and a service repository which consists of a large number of web services associated with randomly generated inputs and outputs. The complexity of the test cases is diverse in terms of the number of web services available and the number of workflow structures involved.

Algorithm 5 The framework for optimization of many (conflicting) objectives

Initialize multiple reference points R, and a random population of size N
$t = 1, i = 1$
while $termination\ criteria\ is\ not\ met$ **do**
  $Q_t \leftarrow selection, crossover\ and\ mutation\ on\ P_t$
  $R_t \leftarrow P_t \cup Q_t$
  Calculate and normalize each objective value for every sc in R_t
  (F_1, F_2, ...) = non-dominated-sort(R_t)
  while $\|S_t\| < N$ **do**
    $S_t = S_t \cup F_i$ and $i = i + 1$
  end while
  if $\|S_t\| = N$ **then**
    $P_{t+1} = S_t$
  else
    Find the reference point R_{min} having the smallest number of associated sc
    Choose ($N - \|S_t\| - \|F_l\|$) sc that are associated with R_{min} from the last front F_l to construct P_{t+1}
  end if
  $t = t + 1$
end while

4.3.2 Parameter Configurations

To simulate the time and cost for data access and transfer in our experiments, the amount of input data for an atomic data-intensive service is randomly generated in the interval (0, 30]MB, and the amount of output data is determined by multiplying by a random factor of 0∼10. For the sake of simplicity, the queue time required by a service server is between 0∼10s, all servers' data write speed (i.e., $ws(s_i)$) and process rate (i.e., $pr(s_i)$) are both 10MB/s, the transfer rate between two service servers (i.e., $bw(s_j, s_i)$) is 3MB/s, and all other costs such as data access cost and transfer cost for per unit of data are all 1. For the extra parameters of F-MOGP, the population size is 200 and the maximum number of generations is 500. The crossover probability is 0.9 and the mutation probability is 1/200 = 0.05. Our approach was repeated 30 times independently on each test case, and the performance statistics (z-test with a 95% confidence level) for the 30 runs are reported.

4.3.3 Experimental Results and Analysis

Search Space Reduction

In order to evaluate the performance of our search space reduced algorithm on the datasets with a large number of web services, we applied F-MOGP to all the test cases supplied by WSC2008 and WSC2009. The number of web services before and after the search space reduction for every test case are reported in Table 4.1. It can be observed that the search space can be significantly reduced by our algorithm. For example, the number of web services was reduced from 608 to 64 for the test case WSC2008-2, which in turn saves considerable time for the following optimization process as indicated by Table 4.1.

Clearly, Table 4.1 shows that all the test cases present a vast original search space and the evaluation of each candidate solution including infeasible solutions requires a significant amount of computing time which

Table 4.1: The number of web services before and after search space reduction for every test case

Test case	Before	After	Test case	Before	After
WSC2008-1	208	192	WSC2008-8	8169	91
WSC2008-2	608	64	WSC2009-1	622	68
WSC2008-3	654	111	WSC2009-2	4179	200
WSC2008-4	1091	80	WSC2009-3	8188	158
WSC2008-5	1140	78	WSC2009-4	8352	162
WSC2008-6	3148	407	WSC2009-5	15261	214
WSC2008-7	4163	52	–	–	–

Table 4.2: The computation time (ms) required for every test case with and without search space reduction (↑ denotes significantly better)

Test case	Without search space reduction	With search space reduction
WSC2008-1	5039 ± 67	$2333 \pm 54\uparrow$
WSC2008-2	13299 ± 46	$2172 \pm 40\uparrow$
WSC2008-3	249728 ± 3605	$176596 \pm 2173\uparrow$
WSC2008-4	10436 ± 93	$2359 \pm 64\uparrow$
WSC2008-5	48056 ± 161	$7422 \pm 43\uparrow$
WSC2008-6	329196 ± 9368	$58286 \pm 415\uparrow$
WSC2008-7	419786 ± 7381	$22069 \pm 332\uparrow$
WSC2008-8	839967 ± 15665	$39782 \pm 543\uparrow$
WSC2009-1	828398 ± 83	$5023 \pm 45\uparrow$
WSC2009-2	397962 ± 4695	$24086 \pm 199\uparrow$
WSC2009-3	625619 ± 11685	$19442 \pm 187\uparrow$
WSC2009-4	4599843 ± 30595	$105578 \pm 2953\uparrow$
WSC2009-5	3044367 ± 14619	$93458 \pm 781\uparrow$

cannot give us a realistic computation time. For example, for the test case WSC2009-4 in Table 4.2, it takes approximate *76* minutes to find a near optimal service composition in searching in the original space. In contrast, by reducing the search space, the computation time has been significantly reduced to *2* minutes. In summary, with the previous search space reduction stage, there is a two-fold reduction in the volume of the search space for the worst case. Meanwhile, for most of the test cases, the computation time has dropped at least *50%*, and even more than an order of magnitude (over 10 times) for 7 out of 13 test cases.

Many-Objective Optimization

Next, we provide more insight into the performance of the service compositions found when four conflicting quality objectives are considered. Figure 4.2 shows the distribution of the evolved service compositions for the test case WSC2009-4 as an example to demonstrate the effectiveness of the proposed F-MOGP approach. This figure is a scatter plot matrix containing all the pairwise scatter plots of the four objectives. The service compositions obtained by a single-objective GP-based approach [93] which transforms the four quality objectives into a single-objective fitness function are also plotted in this figure (as +) for comparison purposes. Due to the page restriction, we do not present the experimental results for the other test cases in this chapter as they all repeat a similar pattern like the one seen in Figure 4.2.

It can be seen from Figure 4.2 that the service compositions found by F-MOGP cover a much wider range and dominate the solutions obtained by the single-objective GP-based approach. As shown in the figure, there is a strong correlation between latency and execution cost, i.e., latency can be increased but with substantial deterioration on execution cost, and vise versa. In addition, the measures of availability and accuracy will be affected by the changes of latency and execution cost. This observation clearly suggests that users could pay more to get better data-intensive services with higher availability and accuracy. Another interesting observation from this figure is the trade-offs between availability and accuracy. The correlation is only obvious when the value of accuracy is low, that is, when the value of accuracy becomes large, it is possible to increase accuracy further without significant deterioration on availability. This makes sense since a high quality service with high accuracy is potential to deliver high availability.

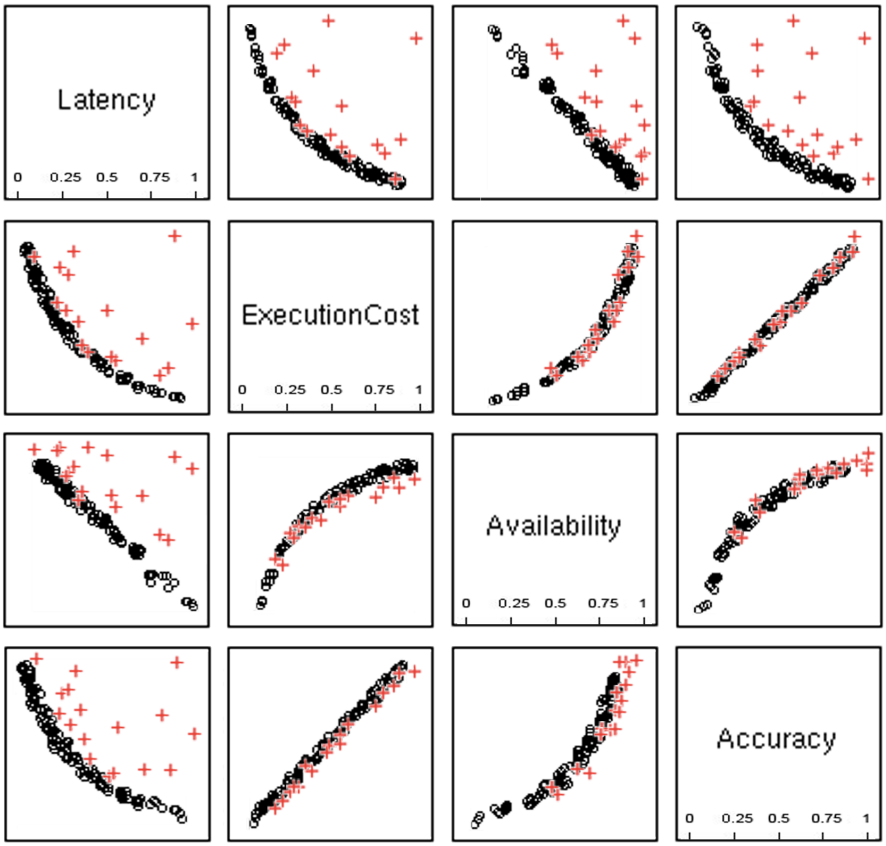


Figure 4.2: Distribution of service compositions on the evolved Pareto front for WSC2009-4

Further Analysis

The preceding experimental studies indicate that the proposed approach is efficient and effective in evolving towards a set of Pareto-optimal service compositions and decision makers would benefit greatly from understanding the potential trade-offs in the evolved Pareto-front. In this section, the performance of two well-known EMO algorithms i.e., NSGA-II [19] and SPEA2 [103], is compared with our approach on the same four-objective problem to further demonstrate the superiority of F-MOGP in tackling many-objective QoS-aware data-intensive service composition.

To investigate the performance of different optimization approaches, a performance metric, i.e., the inverse generational distance (IGD) [20], is employed to provide a combined information about the convergence and diversity of the obtained service compositions. Since the ideal targeted Pareto-front is known in our problems, IGD is used to measure the average distance from the set of non-dominated points S obtained by each approach to the known set of Pareto-optimal points T in the objective space, as described in Equation 4.3.

$$IGD(S,T) = \frac{1}{|T|}\sum_{i=1}^{|T|} min_{j=1}^{|S|} d(t_i, s_j) \tag{4.3}$$

where $d(t_i, s_j)$ is the Euclidean distance between the point t_i in the targeted Pareto-front and the point s_j in the set of non-dominated points found by the approach. Clearly, a smaller IGD value represents a set of higher-quality service compositions. For a meaningful comparison, the parameter settings for different approaches are the same, and the results are obtained after 30 independent runs. The comparisons, in particular, are performed based upon the shrinked benchmark datasets the size of which has been reduced utilizing the search space reduction algorithm discussed in Section 4.4.

Table 4.3 reports the best, mean and worst IGD values for each approach. In order to draw sound conclusions, tests for statistical signifi-

Table 4.3: Best, mean and worst IGD values for every test case obtained by F-MOGP, NSGA-II and SPEA2. Best performance is shown in bold and ↑ denotes significantly better

Test case		F-MOGP	NSGA-II	SPEA2
WSC2008-1	best	3.751×10^{-4}	2.674×10^{-2}	$\mathbf{3.465 \times 10^{-4}}$
	mean	$\mathbf{5.303 \times 10^{-4}}$	2.873×10^{-2}	5.362×10^{-4}
	worst	8.881×10^{-4}	4.315×10^{-2}	$\mathbf{8.859 \times 10^{-4}}$
WSC2008-2	best	$\mathbf{6.132 \times 10^{-4}}$	7.388×10^{-2}	9.761×10^{-4}
	mean	$\mathbf{7.406 \times 10^{-4}}\uparrow$	7.474×10^{-2}	1.097×10^{-3}
	worst	$\mathbf{9.846 \times 10^{-4}}$	7.658×10^{-2}	2.914×10^{-3}
WSC2008-3	best	$\mathbf{8.751 \times 10^{-4}}$	6.119×10^{-2}	9.877×10^{-4}
	mean	$\mathbf{5.028 \times 10^{-3}}\uparrow$	1.555×10^{-1}	7.222×10^{-3}
	worst	$\mathbf{6.985 \times 10^{-3}}$	6.312×10^{-1}	8.913×10^{-3}
WSC2008-4	best	$\mathbf{6.129 \times 10^{-4}}$	1.455×10^{-1}	4.086×10^{-3}
	mean	$\mathbf{5.213 \times 10^{-3}}$	1.505×10^{-1}	5.360×10^{-3}
	worst	9.147×10^{-3}	1.652×10^{-1}	$\mathbf{8.394 \times 10^{-3}}$
WSC2008-5	best	$\mathbf{3.179 \times 10^{-3}}$	2.903×10^{-1}	3.530×10^{-3}
	mean	$\mathbf{3.837 \times 10^{-3}}$	3.194×10^{-1}	3.845×10^{-3}
	worst	$\mathbf{5.189 \times 10^{-3}}$	4.481×10^{-1}	1.697×10^{-2}
WSC2008-6	best	$\mathbf{2.255 \times 10^{-3}}$	3.945×10^{-1}	4.262×10^{-3}
	mean	$\mathbf{3.978 \times 10^{-3}}\uparrow$	7.269×10^{-1}	4.357×10^{-3}
	worst	$\mathbf{8.049 \times 10^{-3}}$	9.046×10^{-1}	1.123×10^{-2}
WSC2008-7	best	9.749×10^{-4}	1.082×10^{-1}	$\mathbf{8.857 \times 10^{-4}}$
	mean	3.273×10^{-3}	4.816×10^{-1}	$\mathbf{3.188 \times 10^{-3}}$
	worst	$\mathbf{5.083 \times 10^{-3}}$	6.434×10^{-1}	7.168×10^{-3}
WSC2008-8	best	$\mathbf{2.058 \times 10^{-3}}$	3.092×10^{-1}	5.294×10^{-3}
	mean	$\mathbf{3.970 \times 10^{-3}}\uparrow$	8.277×10^{-1}	5.937×10^{-3}
	worst	$\mathbf{5.316 \times 10^{-3}}$	9.235×10^{-1}	8.076×10^{-3}
WSC2009-1	best	9.116×10^{-4}	8.361×10^{-2}	$\mathbf{8.047 \times 10^{-4}}$
	mean	$\mathbf{1.208 \times 10^{-3}}\uparrow$	8.728×10^{-2}	1.794×10^{-3}
	worst	$\mathbf{1.880 \times 10^{-3}}$	1.097×10^{-1}	3.669×10^{-3}
WSC2009-2	best	$\mathbf{2.553 \times 10^{-3}}$	2.092×10^{-1}	9.849×10^{-3}
	mean	$\mathbf{6.807 \times 10^{-3}}\uparrow$	3.384×10^{-1}	2.018×10^{-2}
	worst	$\mathbf{4.083 \times 10^{-2}}$	4.321×10^{-1}	4.322×10^{-2}
WSC2009-3	best	$\mathbf{3.624 \times 10^{-3}}$	3.496×10^{-1}	2.097×10^{-2}
	mean	$\mathbf{3.973 \times 10^{-3}}\uparrow$	5.677×10^{-1}	2.468×10^{-2}
	worst	$\mathbf{5.346 \times 10^{-3}}$	8.849×10^{-1}	3.188×10^{-2}
WSC2009-4	best	$\mathbf{3.384 \times 10^{-3}}$	6.336×10^{-1}	6.028×10^{-2}
	mean	$\mathbf{5.479 \times 10^{-3}}\uparrow$	9.677×10^{-1}	8.676×10^{-2}
	worst	$\mathbf{1.366 \times 10^{-2}}$	1.415	1.228×10^{-1}
WSC2009-5	best	$\mathbf{5.751 \times 10^{-3}}$	5.198×10^{-1}	1.733×10^{-1}
	mean	$\mathbf{7.264 \times 10^{-3}}\uparrow$	8.713×10^{-1}	1.949×10^{-1}
	worst	$\mathbf{1.176 \times 10^{-2}}$	9.828×10^{-1}	2.141×10^{-1}

cance (z-test with a 95% confidence level) is further conducted on the average IGD values obtained by the three approaches for every test case. The performance of our F-MOGP approach is significantly better than NSGA-II on all the test cases and SPEA2 on 9 out of 13 test cases. For the remaining test cases (i.e., WSC2008-1, WSC2008-3, WSC2008-6 and WSC2008-7), F-MOGP present similar performance compared with SPEA2. However, in some cases (e.g., WSC2008-3), F-MOGP still obtains the smallest average IGD value without showing statistical significance. In addition, NSGA-II consistently exhibits poor performance for all the test cases indicated by the relatively large IGD values. Since the IGD metric provides a combined information about convergence and diversity, the results in Table 4.3 clearly show that our F-MOGP approach offers the best overall performance for solving many-objective QoS-aware data-intensive service composition compared with state-of-the-art methods.

4.4 Summary

In this chapter, QoS-aware data-intensive service composition is modeled as a many-objective optimization problem and we present a novel filter-based many-objective evolutionary approach named *F-MOGP* to finding a set of Pareto-optimal data-intensive service compositions. To the best of our knowledge, this is the first attempt to apply a many-objective optimization algorithm to this problem. The experimental results show that the computation time of our approach is reduced dramatically by applying the search space reduction algorithm as the first step in the whole process. The service compositions found by our approach are also helpful to support decision makers by providing them with interesting trade-offs among different objectives. The further comparisons demonstrate that the solutions produced by our approach have better quality than the ones produced by existing single-objective and multi-objective approaches.

Chapter 5

A Genetic Programming Approach to Decentralized Execution of Data-Intensive Web Service Compositions

The problem of *"QoS-aware data-intensive web service composition"* has been widely studied in the previous chapters where a hybrid GP-Tabu approach and a many-objective evolutionary approach were proposed to efficiently compose a set of atomic data-intensive services into a composite web service. With the emergence of web services technology, data-intensive applications can be designed and built from creating composite data-intensive web services. Today, however, the execution of these composite services still faces the issues of performance, throughput and scalability. This chapter is therefore concerned with efficient execution of such data-intensive applications.

5.1 The Problem

Typically, a composite web service is executed by a single coordinator node which receives service user's request, makes necessary data transformations and invokes each component service. As the name implies, the coordinator node is responsible for the coordination of all the data and control flows between the component services, and hence becomes a potential performance bottleneck. This execution paradigm of composite web services is referred to as *centralized orchestration* [14]. As depicted in Figure 5.1, all data is transferred between various component services via the coordinator node instead of being transferred directly from the point of generation to the point of consumption, which may cause unnecessary network traffic overloading. In addition, it is possible for a component service to produce a lot of data that is irrelevant to the desired function of the composite web service. However, the data will be still transferred to the coordinate node, thereby resulting in increased workload on the network as well as the increase of response time. This is particularly problematic for data-intensive service compositions in which huge collections of data are transferred.

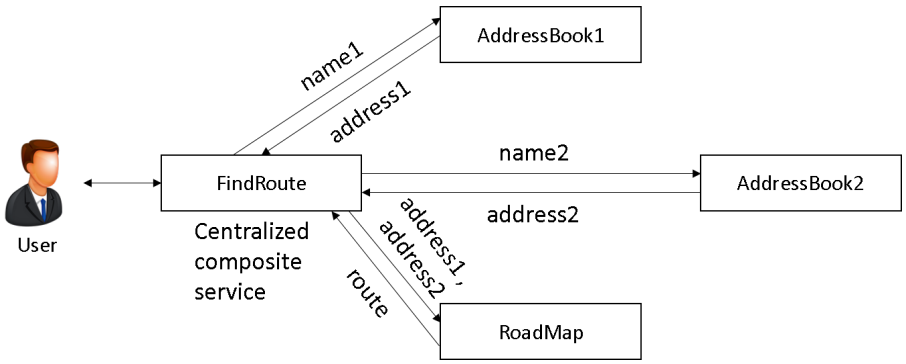


Figure 5.1: An example of centralized orchestration

In summary, such a highly centralized execution paradigm could lead to several potential drawbacks.

- The execution of a composite web service generates a large volume of unnecessary network traffic due to the fact that all intermediate results have to be sent back to the central workflow coordinator before the following web services can be invoked. In most cases of data-intensive processes, this consumes valuable bandwidth and causes a performance bottleneck.

- The scalability and reliability of this centralized execution paradigm cannot be guaranteed as there can be a single point of failure.

- The overall performance of a composite web service can be affected by the number of service requests on the central workflow coordinator and available resource capacity of the server.

As one of the most popular languages for web service compositions, Business Process Execution Language (BPEL) [5] is often adopted to define how a composite web service will be executed by expressing a composite web service as an end-to-end process flow. A simple business process written in the BPEL language is described in Figure 5.2. In this work, the problem of distributed execution of data-intensive web service compositions is regarded as the problem of partitioning data-intensive BPEL processes where a data-intensive BPEL process is partitioned into several sub-processes which will be deployed and executed on separate workflow engines, in order to improve the performance of the whole workflow.

The problem of partitioning BPEL processes was firstly introduced by Nanda et al. [61] as follows.

- A BPEL process consists of a set of statements (i.e., activities) which can be categorized into fixed activities and mobile activities. A fixed activity such as $receive$, $reply$ and $invoke$ must be deployed and executed on a particular workflow engine, while a mobile activity such as $send$, $assign$ and if can be assigned to any workflow engine.

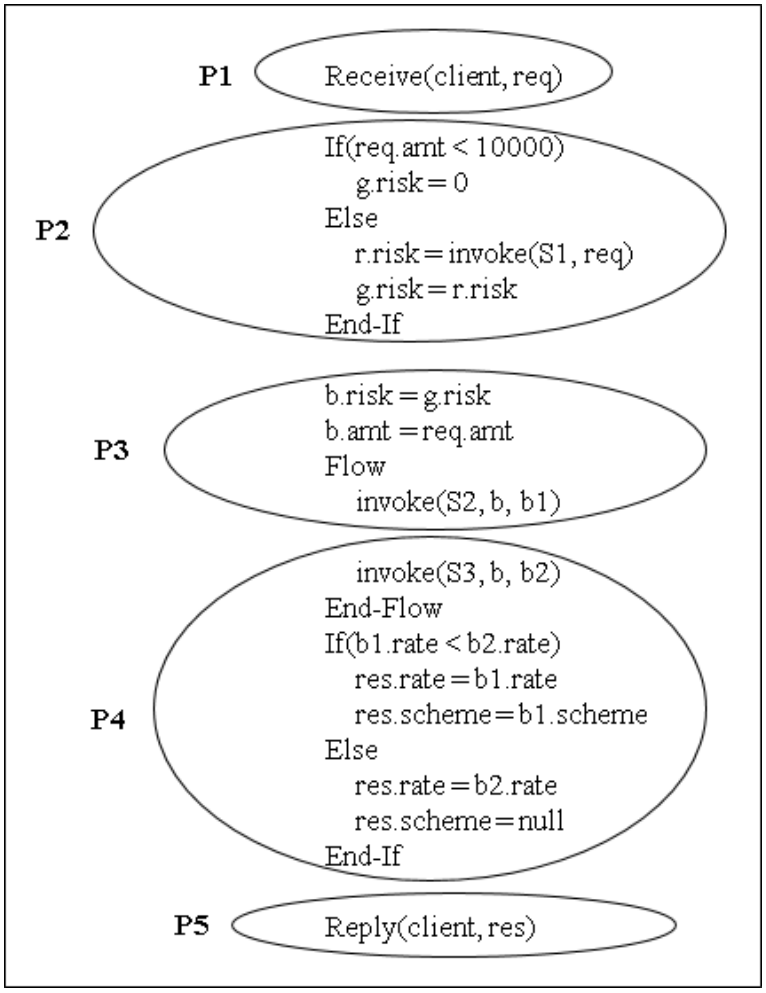


Figure 5.2: An example of a business process expressed in the BPEL language

- There are two different types of dependence, i.e., data dependence and control dependence, between two activities. Control dependence is a situation in which an activity executes if the previous activity evaluates in a way that allows its execution. For example, as shown in Figure 5.2, the activity *"g.risk=0"* is control dependent on *"if(req.amt <10000)"* as the execution of the first activity is conditionally guarded by the second one. A data dependence is concerned about two activities which access or modify the same resource. For example, the activity *"if(req.amt<10000)"* is flow dependent on *"Receive(client, req)"* since the first activity needs the data from the second one which has execution precedence. In addition, activities with different control dependence cannot be placed into the same partition unless their control activities are in the same partition. In order to guarantee that all the BPEL sub-processes can be coordinated correctly to perform the overall process execution, the preceding dependencies constraints must be properly preserved by the partitions.

- A partition of a BPEL process (i.e., a BPEL sub-process) is made up of exactly one fixed activity and zero or more mobile activities.

Based on the above descriptions, the problem of partitioning a BPEL process can be formally defined as a quadruple BPEL = $\{F, M, D, C\}$ where

- F denotes a set of fixed activities $\{f_1, f_2, ..., f_n\}$,

- M denotes a set of mobile activities $\{m_1, m_2, ..., m_m\}$,

- D represents a set of data dependencies existing between any two activities $\{(a_i, a_j) \,|\, a_j$ is data dependent $a_i\}$,

- and C represents a set of control dependencies existing between any two activities $\{(a_i, a_j) \,|$ the execution of a_j is conditionally guarded by $a_i\}$

Given a BPEL process, different partitioning execution plan of the same process will give rise to different execution performance. Finding an optimal distribution (i.e., partitioning) of the process is dependent on a number of influential factors such as communication overhead. Due to the underlying high complexity of a process, the problem of partitioning BPEL processes is an optimization problem with a high-dimensional search space and it has proven to be NP-hard [1]. Therefore, how to effectively partition BPEL processes is still a challenging issue today, especially taking into account the data-intensity involved in a process.

A few heuristics [14, 26, 76, 87] have been put forth to address the above drawbacks of centralized execution paradigm. Most of the research work in this field splits a workflow into several sub-workflows, each of which can be deployed and executed on a separate workflow engine for decentralized execution of a composite web service. However, most of existing proposals do not take data intensity into consideration when splitting a data-intensive workflow (i.e., a data-intensive web service composition). Moreover, these approaches focus on reducing the communication costs only, and their scalability is not evaluated. In this chapter, we present a genetic programming based approach to decentralized execution of data-intensive web service compositions. The main contributions of this chapter are three-fold. First, a mathematical model is developed to describe the communication overhead including latency and costs when partitioning a data-intensive workflow. Second, the semantics (i.e., data and control dependencies) of the original workflow can be properly preserved by our approach. Last, our approach proves to have better scalability than two existing methods based on experimental observations.

The rest of this chapter is organised as follows. Section 5.2 presents the model used to estimate the communication overhead during the partitioning process of a data-intensive workflow. In Section 5.3, the details of the proposed approach are explained, and Section 5.4 reports on the experimental results. Finally, the conclusions are drawn in Section 5.5.

5.2 A Model for Estimating Communication Overhead

The execution plans of partitioned data-intensive BPEL processes are expected to have the least communication overhead in terms of latency and costs. During the execution process, communication overhead could take place inside a partition and across two partitions. A mathematical model is therefore needed to estimate the communication overhead under the two circumstances mentioned above.

5.2.1 Inside A Partition

To measure the data communication overhead inside a partition, we need to work out the sum of the communication overhead produced by all the $invoke$ activities within the partition. In other terms, as shown in Equation 5.1, the communication overhead is equal to the sum of the total latency and total costs due to mass data transfer between the partition and the data-intensive web services to be invoked by the partition.

$$communication\_overhead = \sum_{i=1}^{j}(latency_i + costs_i) \tag{5.1}$$

where j denotes the total number of invoke activities in the partition, and $latency_i$ and $costs_i$ can be calculated using Equations 5.2 and 5.5 respectively.

$$latency = numExec(a) * (q + p(d_s) + t(d_s, p, s) + t(d_r, s, p)) \tag{5.2}$$

where q is the time spent in waiting in the queue for the service s to be invoked, $p(d_s)$ is the actual time used by the service to process the set of data d_s, $t(d_s, p, s)$ is the data transfer time for transferring the data set d_s from the partition p to the service s, and $t(d_r, s, p)$ is the data transfer time

for transferring the processed data set d_r from the service s back to the partition p. Since an activity may be executed more than once, the execution times of a particular invoke activity a, i.e., $numExec(a)$, depends on its parent control-flow patterns. If a is involved in a sequence or parallel control-flow, then it will be executed exactly once. However, if a is involved in a choice control-flow which has a branching probability of p, then the activity will be executed p times. For a loop control-flow that has a repeat probability of q, the execution times of the activity a will be $1/(1-q)$. Therefore, the average execution times of the activity a is determined by the probabilities of the choice and loop control-flows which appear in the path from the root of the BPEL process to a. Note that control-flow patterns can be nested so when a choice control-flow with probability p is traversed, the number of executions of a is multiplied by p, while a loop control-flow with probability q is traversed the number of executions is multiplied by $1/(1-q)$.

For the remaining parts of Equation 5.2, the queue time q depends on the server load, i.e., the current request queue length on the server, the processing time p depends on the processing capacity of the server that provides the service, and the data transfer time t is determined by the network bandwidth and the amount of data to be transferred between two servers, as defined in Equations 5.3 and 5.4.

$$p(d) = size(d)/pr(s) \tag{5.3}$$

$$t(d, x, y) = size(d)/bw(x, y) + size(d)/ws(y) \tag{5.4}$$

where $size(d)$ is the size of the data set d, $pr(s)$ denotes the processing rate of the server hosting the service s, $bw(x, y)$ is the network bandwidth between the server x and the server y, and $ws(y)$ is the disk write speed of the server y.

The communication costs inside a partition is the sum of data access cost, data transfer cost, and service related cost as shown in Equation 5.5. The data access cost $ac(d_s, s)$ is the price to be paid for writing the data

d_s to the server that hosts the service s and reading the data in order to invoke the service. The data transfer cost tc is proportional to the size of the data set, which depends on the available network bandwidth between two servers. The service related cost sc expresses the cost to provision the service s as well as the cost to process the service request including data processing.

$$cost = numExec(a) * (sc(s) + tc(d_s, p, s) + ac(d_s, s) + tc(d_r, s, p) + ac(d_r, p)) \tag{5.5}$$

$$ac(d, y) = size(d) * wcost(y) + size(d) * rcost(y) \tag{5.6}$$

$$tc(d, x, y) = size(d) * tcost(x, y) \tag{5.7}$$

$$sc(s) = pcost(s) + size(d) * dcost(s) \tag{5.8}$$

where $size(d)$ denotes the size of the data set d, $wcost(y)$ is the cost of writing per unit of data to the server y, $rcost(y)$ is the cost of reading per unit of data from the disk on the server y, $tcost(x, y)$ is the transfer cost from the server x to the server y for per unit of data, $pcost(s)$ is the price charged to use the service s which is usually specified by service provider, and $dcost(s)$ is used to represent the expenditure for processing each unit of data on the server hosting the service s.

5.2.2 Between Two Partitions

As the size of the messages carrying control-flows information between two partitions is small in a data-intensive process, we only consider the data communication overhead when there is a data dependence between two partitions. The equations are defined in a similar way as those in 5.2.1 except that the transfer time and costs need to be calculated only once due to the unidirectional data-flow. Due to the similarities, the details of estimating the communication overhead between pairs of partitions are not discussed in this chapter.

5.3 The GP-Based Partitioning Approach for Data-Intensive Processes

In this chapter, we propose a GP-based approach to partitioning a BPEL data-intensive process so as to overcome the shortcomings of centralized execution paradigm. Our proposed approach follows a standard genetic programming framework which has been described in Chapter 2. The approach starts with taking a BPEL process workflow and information about the data and control dependencies between pairs of activities as inputs, as presented in Algorithm 6. Then, a certain number of partition solutions are randomly produced to form an initial population. This is done by randomly assigning every fixed activity to a partition and every mobile activity to a fixed activity. The quality of each candidate solution in the population is evaluated based on the fitness function explained in Section 5.3.4. Afterwards, two partition solutions are stochastically selected from the current population to perform crossover and mutation in order to generate new offsprings for next generation. The preceding steps are repeated until the maximum number of generations, i.e., g_{max}, have been reached. Eventually, the output of the algorithm is a single optimal partition solution to a given data-intensive BPEL process.

5.3.1 Representation

In our approach, a partitioning execution plan for a BPEL process workflow is represented as a tree structure, as depicted in Figure 5.3. The function set is made up of fixed activities denoted by rectangular nodes in the tree, and the terminal set consists of mobile activities denoted by circle nodes. Each function node can have zero or more terminals as its children which means that the fixed activity and its children (i.e., mobile activities) are assigned to the same partition. To capture the executional ordering of the activities within a BPEL program, each node in the tree is associ-

Algorithm 6 A GP-based algorithm for splitting a BPEL process for optimally decentralized execution

Require: a data-intensive BPEL process, data and control dependencies information in the process

Ensure: a feasible and optimal partitioning execution plan

1: Set g = 1 // the current generation
2: Generate an initial population P filled with partition solutions randomly
3: Evaluate the quality of each partition solution in the population using the fitness function
4: **while** $g < g_{max}$ **do**
5: **while** $|P'| < |P|$ **do**
6: Select two candidate partition solutions from the current population P
7: Perform crossover and mutation on the selected solutions and add the offsprings produced to the new population P'
8: **end while**
9: Evaluate each partition solution in P' using the fitness function
10: **end while**
11: **return** partitioning execution plan having the best fitness value

ated with a line number indicating the position of the activity in the original BPEL program. In addition, each node stores the information about its data and control dependencies which are visualized by solid lines and dotted lines, respectively in Figure 5.3.

According to the requirements of the BPEL program partitioning problem defined in Section 5.2, a feasible partitioning execution plan such as the one illustrated in Figure 5.3 must guarantee that activities with different control dependence cannot be assigned into the same partition unless their control activities are in the same partition. Also, there exists no cyclic data dependence among the partitions.

5.3.2 Selection

Tournament selection [57] as one of the most popular selection methods in GP, is adopted by the approach to select individuals from a population to generate new offsprings for next generation. This method holds a tournament among k individuals which are randomly selected from the original population. The winner of the tournament, i.e., the individual with the highest fitness among the k tournament competitors, is used to form the basis of the next generation. In general, tournament selection is expected to produce more diverse populations [57].

5.3.3 Crossover and Mutation

Crossover and mutation as the main genetic operators in GP are performed on the individuals selected by the selection method described in the previous section, so as to produce the next generation. Figure 5.4 shows an example of the crossover operation where two mobile activities in the two trees are randomly selected and they are swapped between the two trees.

To apply the mutation operator, as illustrated in Figure 5.4, a mobile activity in the tree is randomly selected. The activity is then moved from the

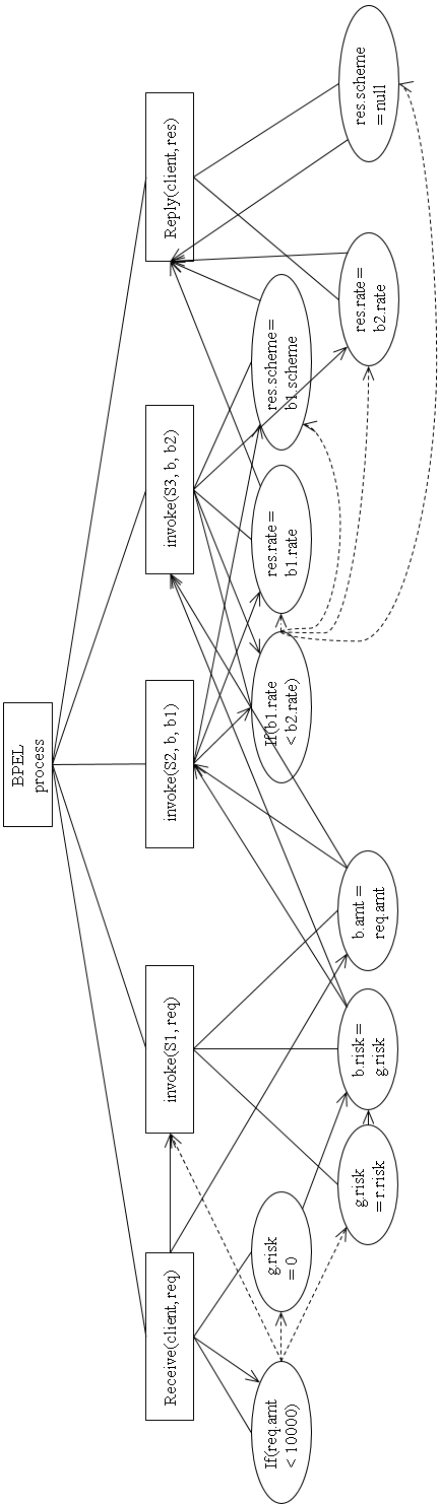


Figure 5.3: A tree-based representation of a partitioning execution plan showing data and control dependencies

fixed activity which it currently belongs to, to another randomly selected fixed activity.

5.3.4 Fitness Function

A fitness function in GP is used to measure how close a particular solution is to the problem's required outputs. In this work, the fitness function mainly focuses on the measurement of the communication overhead incurred during the execution of a partitioning plan. As mentioned previously, our approach starts with potentially infeasible solutions and we need to guide the search towards feasible solutions. In fact, exploiting infeasible search space tends to yield an optimal solution more efficiently. Therefore, the approach allows the movement over infeasible regions of the search space but applies a penalty to their fitness values. The complete fitness function is shown in Equation 5.9 where $num(d)$ and $num(c)$ reveal the quantities of violations on data and control dependencies of a particular partitioning execution plan. It can be noted that an infeasible solution which violates more data and control dependencies will be penalized more.

$$fitness = \frac{1}{communication} + penalty * (num(d) + num(c)) \tag{5.9}$$

where $communication$ represents the communication overhead during the execution process of a partitioning plan, and this can be estimated according to the model proposed in Section 5.3.

A tree is a particularly natural structure for representing a BPEL process, which also simplifies the checking for data and control dependencies violations by a partitioning execution plan. According to the data and control dependencies information obtained from the original BPEL program, the dependency constraints can be verified easily through traversing the tree.

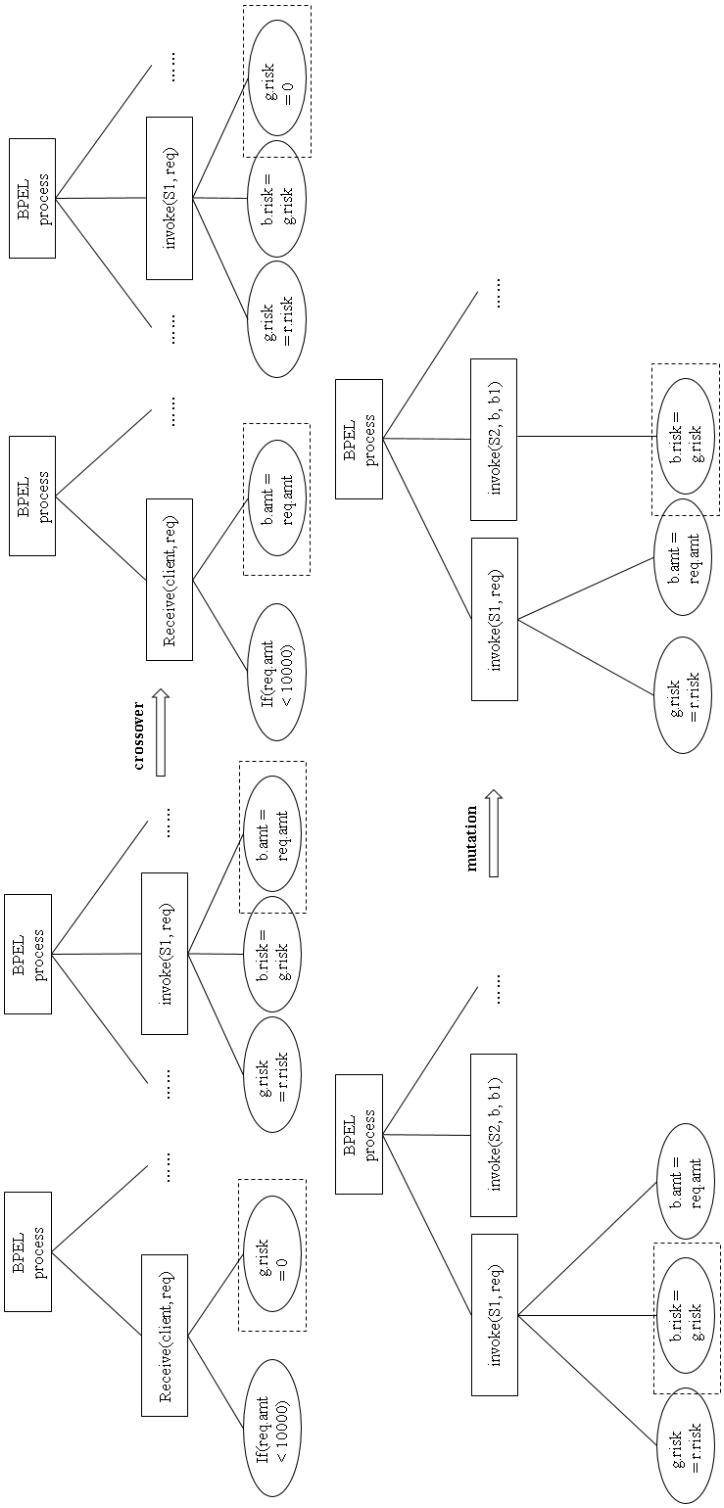


Figure 5.4: The examples for how to perform crossover and mutation in our GP-based approach

As it can be observed from Equation 5.9, the first component of the fitness function is expected to be maximized while the second component is expected to be minimized. That is, the BPEL program partitioning problem is converted to a maximum optimization problem, the objective of which is to find a feasible partitioning execution plan that has the maximum fitness value.

5.4 Experimental Studies

In order to assess the performance of the approach presented in this chapter, we conduct a set of experiments based upon a number of randomly generated data-intensive BPEL processes. Our approach is also compared with two existing methods named MDU [61] and PGM [61] to further demonstrate the scalability of the approach.

5.4.1 Test Cases

To the best of our knowledge, there exists no benchmark tool or framework for the problem of partitioning data-intensive BPEL processes based on EC approaches. Therefore, we generate 8 test cases randomly each of which is a data-intensive BPEL process. Table 5.1 summarizes the hypothetical data-intensive BPEL processes used to evaluate the proposed approach and they have been designed to cover various degree of complexity. As can been seen in Table 5.1, each test case contains different number of fixed activities and mobile activities, as well as different number of control branches. The complexity of the process increases as the number of activities and the level of control dependencies increase. The rationale behind such a design is to demonstrate the effectiveness of the approach regardless of the complexity.

Table 5.1: Hypothetical test cases used in the experiments.

Test case	No. of control branches	No. of fixed activities	No. of mobile activities
1	0	6	6
2	1	15	13
3	2	19	16
4	3	22	19
5	4	31	24
6	5	56	43
7	6	63	54
8	7	82	67

5.4.2 Parameter Configurations

The experiments are conducted on a personal computer with 3.0 GHz CPU and 3.7 GB RAM. The population size of GP is 100, and the maximum number of generations g_{max} is 200. The crossover probability is 0.9 and the mutation probability is 0.01 as we found that this combination can produce good results.

To calculate the communication overhead presented in the model in Section III, all necessary variables used in the calculations are generated in our experiments as follows. The amount of input data for an atomic data intensive service is randomly generated in the interval (0, 300]MB, and the amount of output data is determined by multiplying by a random factor of 0∼10. For the sake of simplicity, the queue time required by a service server is between 0∼10s, all server's data write speed (i.e., $ws(s)$) and process rate (i.e., $pr(s)$) are both 10MB/s, the transfer rate between any two servers (i.e., $bw(x, y)$) is 3MB/s, and all other costs such as data access cost and transfer cost for per unit of data are all 1.

5.4.3 Experimental Results and Analysis

To compare the performance of the three approaches, each approach is applied on each test case shown in Table 5.1. Since our approach is non-deterministic, 30 independent runs are performed for each test case. By

contrast, the MDU and PGM algorithms are only performed once for each test case due to the deterministic nature of the two algorithms. The average computation time and the average best fitness values obtained by each approach for each test case is recorded and presented in Table 5.2.

Table 5.2: The average computation time and the average fitness values obtained by the three approaches for the test cases (↑ denotes significantly better).

Test case	GP		MDU		PGM	
	time(s)	fitness	time(s)	fitness	time(s)	fitness
1	23.31	0.818 ± 0.023	2.14	0.886 ± 0.000	1.52	0.869 ± 0.000
2	27.84	0.815 ± 0.025	3.31	0.865 ± 0.000	2.44	0.865 ± 0.000
3	38.92	$0.824 \pm 0.012\uparrow$	26.77	0.813 ± 0.000	17.39	0.813 ± 0.000
4	43.76	$0.843 \pm 0.026\uparrow$	796.28	0.758 ± 0.000	75.67	0.758 ± 0.000
5	61.09	$0.744 \pm 0.017\uparrow$	–	–	477.61	0.701 ± 0.000
6	87.98	$0.757 \pm 0.009\uparrow$	–	–	–	–
7	139.42	$0.813 \pm 0.022\uparrow$	–	–	–	–
8	154.18	$0.712 \pm 0.016\uparrow$	–	–	–	–

As it can be observed from Table 5.2 and Figure 5.5, the computation time required by MDU and PGM exhibits exponential growth as the complexity of the test case increases. For the first three test cases, the PGM algorithm spends the least computation time in finding a solution among the three approaches. The MDU algorithm also spends much less computation time than our approach in finding a solution. However, starting from test case 4 with 22 fixed activities, 19 mobile activities and 3 control branches, both MDU and PGM require more time than our approach. For example, our approach spends $43.76s$ in searching for an optimal solution but the MDU algorithm needs $796.28s$. When the test case becomes more complex, neither the MDU algorithm nor the PGM algorithm is able to find a solution within a reasonable time period. In contrast, our GP-based approach is always able to find a solution within $155s$ for all the test cases and the computation time depicted in Figure 5.5 presents a slow increase as the complexity of the test case increases.

In addition, we also apply the fitness function used by our approach

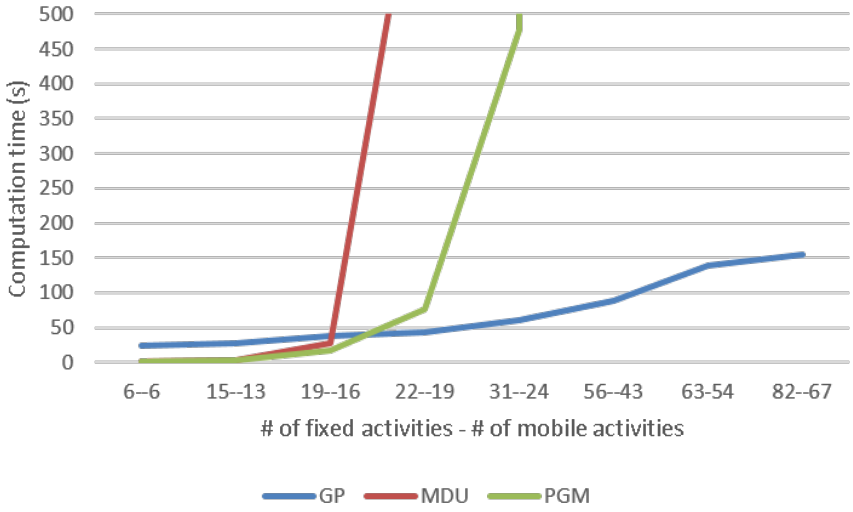


Figure 5.5: The comparisons of the average computation time used by our GP-based approach, MDU and PGM for the test cases

to the solutions found by the MDU and PGM algorithms for a meaningful comparison of the qualities of the solutions found by the three approaches. In order to draw sound conclusions, tests for statistical significance (z-test with a 95% confidence level) is further conducted on the average fitness values obtained by the three approaches for every test case. As shown in Table 5.2, MDU and PGM can find better solutions than our approach for the first two simple test cases. However, for the remaining test cases in which the processes are complex, our GP-based approach outperforms both MDU and PGM implied by the significant improvement on the qualities from the statistical test.

The analysis of the preceding experimental results indicates that our proposed approach exhibits better scalability than the other two existing approaches. In practice, our approach is recommended for complex data-intensive BPEL processes partitioning problems. However, the MDU and PGM algorithms are still preferred for simple partitioning problems due to their quick convergence and high-quality solutions.

5.5 Summary

Traditional centralized orchestration of composite web services delegates the responsibility for coordinating the execution of a composite service to a single workflow engine, which results in potential performance bottleneck and a single point of failure. Such problems could be resolved by decentralizing execution of composite web services, however, this raises the issue of how to partition a composite service in an efficient and effective manner. In this chapter, we propose a genetic programming based approach to realizing decentralized execution of data-intensive web service compositions. This is achieved by splitting a data-intensive BPEL process into a set of sub-processes which will be deployed and executed on separate workflow engines. The data intensity of the process is considered by adopting a mathematical model to estimate the data communication overhead during the decentralized execution process. Compared with two existing heuristic algorithms (i.e., MDU and PGM), the experimental results show the good scalability of our proposed approach and the superiority of the approach over the two algorithms in solving complex problems.

Chapter 6

Conclusions

At present more and more software capabilities are delivered and consumed over the Internet or the Cloud as web services which form a core component of service-oriented architecture. In most cases, instead of a single service, a range of web services are required to be composed together in order to provide new value-added and complex functionality. On the other hand, in the face of tremendous web services offering identical or overlapping functionality, quality of service (QoS) characteristics (e.g., latency, execution cost, availability) need to be considered for service composition. In general, in addition to functional requirements, end users have certain non-functional requirements as well for the resulting composite service. For example, the latency and execution cost of a composite service are preferred to be minimized meanwhile can achieve a relatively high availability and reliability. Furthermore, with the exponential growth of the amount of data generated by humanities, scientific activities, as well as commercial applications from a diverse range of fields, it has been realized that the service-oriented approach using web services is also of great interest for the implementation of data-intensive processes such as data mining and image processing. Due to the sustainable growth of data volumes used in the fields of sciences and engineering, finance, media, online information resources and so on, *data-intensive service composition* has be-

come the foremost research area in academia and industry. Therefore, the overall goal of this research work is to find an effective approach to QoS-aware data-intensive web service composition and execution, in order to address the effect of data intensity and realize distributed execution of composite web services. This goal has been successfully achieved by the proposed approaches presented in this thesis.

In this thesis, we firstly proposed a hybrid GP-Tabu approach to solving the QoS-aware data-intensive web service composition problem. In order to cope with the intrinsic data-intensive nature of data-intensive web services, a mathematical model has been proposed to estimate the communication time and costs spent on the data transfer between data-intensive services. The approach was successfully applied to two public benchmark datasets, both of which contain a large number of web services. The experimental results showed that our new approach outperformed the simple GP and two traditional optimization methods (i.e., TS and ILP) in finding more satisfying data-intensive service compositions.

Next, we proposed a many-objective evolutionary approach to the problem of QoS-aware data-intensive web service composition in order to efficiently optimize four conflicting quality dimensions simultaneously and generate a set of Pareto-optimal service compositions when users are not certain about the importance of each quality dimension. A search space reduction strategy was firstly applied to reduce the original search space of the problem. Then a NSGA-III based algorithm was adopted to solve the many-objective optimization problem. Two important evolutionary multi-objective algorithms (i.e., NSGA-II and SPEA2) were compared with our approach based on the same simulation experiments in order to evaluate the effectiveness and efficiency of the proposed approach. We found that our approach was computationally efficient due to the search space reduction, and it was more suitable for many-objective service composition problems with high-dimensional objective space.

Finally, because of the drawbacks of centralized execution paradigm of

composite web services, we proposed a GP-based approach to partitioning a data-intensive BPEL process which represents a data-intensive web service composition, into a set of sub-processes that can be deployed and executed on separate workflow engines for decentralized execution. The data communication overhead between pairs of the resulting partitions was also properly measured according to a mathematical model. The experimental results revealed that the proposed approach offered better scalability than two existing heuristic algorithms (i.e., MDU and PGM), and it could generate better partitioning execution plans with the least communication overhead for large-scale problems.

6.1 Future Work

Future work includes examining the effectiveness and efficiency of the proposed many-objective evolutionary approach on the QoS-aware data-intensive web service composition problems having higher-dimensional objective space (i.e., more than four objectives to be optimized). The approach will also be compared with other powerful algorithms proposed in the literature such as MOEA/D [100] to further demonstrate the superiority of the approach. Furthermore, a decision making process will be developed to assist a decision maker to compare the quality of the service compositions in the Pareto-front evolved and find the best solution to a given service composition problem. Finally, all the approaches presented in this thesis will be applied in real life situations so the variables used for estimating communication time and costs (e.g., queue time, processing time, transfer time, etc.) can be measured with real values so as to demonstrate the performance of the proposed approaches in solving real-world problems.

Bibliography

[1] AI, L., TANG, M., AND FIDGE, C. Partitioning composite web services for decentralized execution using a genetic algorithm. *Future Gener. Comput. Syst. 27*, 2 (2011), 157–172.

[2] AL-MASRI, E., AND MAHMOUD, Q. H. Qos-based discovery and ranking of web services. In *ICCCN 2007. Proc. of 16th International Conference on* (2007), IEEE, pp. 529–534.

[3] AL-MASRI, E., AND MAHMOUD, Q. H. Qos-based discovery and ranking of web services. In *Proceedings of the 16th International Conference on Computer Communications and Networks, IEEE ICCCN 2007, Turtle Bay Resort, Honolulu, Hawaii, USA, August 13-16, 2007* (2007), IEEE.

[4] ALRIFAI, M. Distributed and scalable qos optimization for dynamic web service composition. In *Proceedings of the PhD Symposium at the 6th International Conference on Service Oriented Computing (ICSOC 2008), Sydney, Australia, 1 December 2008* (2008), H. R. M. Nezhad, F. Toumani, and Y. Velegrakis, Eds., vol. 421 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[5] ANDREWS, T., CURBERA, F., DHOLAKIA, H., GOLAND, Y., KLEIN, J., LEYMANN, F., LIU, K., ROLLER, D., SMITH, D., THATTE, S., ET AL. *Business process execution language for web services*. version, 2003.

[6] AVERSANO, L., DI PENTA, M., AND TANEJA, K. A genetic programming approach to support the design of service compositions. *International Journal of Computer Systems Science and Engineering 4* (2006), 247–254.

[7] BANSAL, A., BLAKE, M., KONA, S., BLEUL, S., WEISE, T., AND JAEGER, M. Wsc-08: Continuing the web services challenge. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on* (July 2008), pp. 351–354.

[8] BANZHAF, W., FRANCONE, F. D., KELLER, R. E., AND NORDIN, P. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[9] BATOUCHE, B., NAUDET, Y., AND GUINAND, F. Semantic web services composition optimized by multi-objective evolutionary algorithms. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on* (2010), IEEE, pp. 180–185.

[10] BUCCHIARONE, A., AND PRESTI, L. Qos composition of services for data-intensive application. In *ICIW '07. Second International Conference on* (2007), pp. 46–46.

[11] CANFORA, G., DI PENTA, M., ESPOSITO, R., AND VILLANI, M. L. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation* (New York, NY, USA, 2005), GECCO '05, pp. 1069–1075.

[12] CANFORA, G., DI PENTA, M., ESPOSITO, R., AND VILLANI, M. L. An approach for qos-aware service composition based on genetic

algorithms. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (2005), GECCO '05, ACM, pp. 1069–1075.

[13] CARDELLINI, V., CASALICCHIO, E., GRASSI, V., AND LO PRESTI, F. Flow-based service selection for web service composition supporting multiple qos classes. In *Web Services, 2007. ICWS 2007. IEEE International Conference on* (July 2007), pp. 743–750.

[14] CHAFLE, G. B., CHANDRA, S., MANN, V., AND NANDA, M. G. Decentralized orchestration of composite web services. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters* (New York, NY, USA, 2004), ACM, pp. 134–143.

[15] CHANG, W.-C., WU, C.-S., AND CHANG, C. Optimizing dynamic web service component composition by using evolutionary algorithms. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on* (Sept 2005), pp. 708–711.

[16] CLARO, D. B., ALBERS, P., AND HAO, J.-K. Selecting web services for optimal composition. In *ICWS International Workshop on Semantic and Dynamic Web Processes, Orlando-USA* (2005).

[17] CLARO, D. B., ALBERS, P., AND HAO, J.-K. Selecting web services for optimal composition. In *ICWS International Workshop on Semantic and Dynamic Web Processes, Orlando* (2005).

[18] CREMENE, M., SUCIU, M., POP, F.-C., PALLEZ, D., AND DUMITRESCU, D. A multi-objective approach for qos-aware service composition. 67–74.

[19] DEB, K., AGRAWAL, S., PRATAP, A., AND MEYARIVAN, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. 849–858.

[20] DEB, K., AND JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on 18*, 4 (Aug 2014), 577–601.

[21] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on 6*, 2 (2002), 182–197.

[22] DENG, S., HUANG, L., LI, Y., AND YIN, J. Deploying data-intensive service composition with a negative selection algorithm. *Int. J. Web Service Res. 11*, 1 (2014), 76–93.

[23] DONNET, B., GUEYE, B., AND KAAFAR, M. A. A survey on network coordinates systems, design, and security. *Commun. Surveys Tuts. 12*, 4 (Oct. 2010), 488–503.

[24] DORIGO, M., AND DI CARO, G. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (1999), vol. 2, IEEE.

[25] EL HADDAD, J., MANOUVRIER, M., RAMIREZ, G., AND RUKOZ, M. Qos-driven selection of web services for transactional composition. In *Web Services, 2008. ICWS '08. IEEE International Conference on* (Sept 2008), pp. 653–660.

[26] FDHILA, W., DUMAS, M., AND GODART, C. Optimized decentralization of composite web services. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on* (Oct 2010), pp. 1–10.

[27] GALINIER, P., AND HAO, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization 3*, 4 (1999), 379–397.

[28] GARROPPO, R. G., GIORDANO, S., AND TAVANTI, L. A survey on multi-constrained optimal path computation: Exact and approximate algorithms. *Computer Networks 54*, 17 (2010), 3081–3107.

[29] GLOVER, F., AND LAGUNA, M. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[30] GLOVER, F., AND LAGUNA, M. *Tabu search*. Springer, 1999.

[31] GOLDBERG, D. E. Genetic algorithms in search, optimization and machine learning.

[32] GORGES-SCHLEUTER, M. On the power of evolutionary optimization at the example of atsp and large tsp problems. In *European Conference on Artificial Life. Brighton, UK* (1997), Citeseer.

[33] GROSAN, C., AND ABRAHAM, A. Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In *Hybrid evolutionary algorithms*. Springer, 2007, pp. 1–17.

[34] HORN, J., NAFPLIOTIS, N., AND GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (1994), IEEE, pp. 82–87.

[35] ISHIBUCHI, H., AND MURATA, T. Multi-objective genetic local search algorithm. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on* (1996), IEEE, pp. 119–124.

[36] ISHIBUCHI, H., AND MURATA, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 28*, 3 (1998), 392–403.

[37] ISHIBUCHI, H., YAMAMOTO, N., MURATA, T., AND TANAKA, H. Genetic algorithms and neighborhood search algorithms for fuzzy

flowshop scheduling problems. *Fuzzy Sets and Systems 67*, 1 (1994), 81–100.

[38] ISHIBUCHI, H., YOSHIDA, T., AND MURATA, T. Balance between genetic search and local search in hybrid evolutionary multi-criterion optimization algorithms. In *GECCO* (2002), vol. 2, pp. 1301–1308.

[39] ISHIBUCHI, H., YOSHIDA, T., AND MURATA, T. Selection of initial solutions for local search in multiobjective genetic local search. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on* (2002), vol. 1, IEEE, pp. 950–955.

[40] JAEGER, M., ROJEC-GOLDMANN, G., AND MUHL, G. Qos aggregation for web service composition using workflow patterns. In *EDOC 2004. Proc. Eighth IEEE International* (2004), pp. 149–159.

[41] JAEGER, M. C., AND MUEHL, G. Qos-based selection of services: The implementation of a genetic algorithm. In *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference* (Feb 2007), pp. 1–12.

[42] JASZKIEWICZ, A. Genetic local search for multi-objective combinatorial optimization. *European journal of operational research 137*, 1 (2002), 50–71.

[43] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (1995), vol. 4, IEEE, pp. 1942–1948.

[44] KHALAF, R., KOPP, O., AND LEYMANN, F. Maintaining data dependencies across bpel process fragments. In *Service-Oriented Computing ICSOC 2007*, vol. 4749 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 207–219.

[45] KHALAF, R., AND LEYMANN, F. E role-based decomposition of business processes using bpel. In *Web Services, 2006. ICWS '06. International Conference on* (Sept 2006), pp. 770–780.

[46] KLEIN, A., ISHIKAWA, F., AND HONIDEN, S. Towards network-aware service composition in the cloud. In *Proceedings of the 21st international conference on World Wide Web* (2012), ACM, pp. 959–968.

[47] KONA, S., BANSAL, A., BLAKE, M., BLEUL, S., AND WEISE, T. Wsc-2009: A quality of service-oriented web services challenge. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on* (July 2009), pp. 487–490.

[48] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA, 1992.

[49] LAWLER, E. L., AND WOOD, D. E. Branch-and-bound methods: A survey. *Operations research 14*, 4 (1966), 699–719.

[50] LEE, J. Matching algorithms for composing business process solutions with web services. In *E-Commerce and Web Technologies, 4th International Conference, EC-Web, Prague, Czech Republic, September 2-5, 2003, Proceedings* (2003), K. Bauknecht, A. M. Tjoa, and G. Quirchmayr, Eds., vol. 2738 of *Lecture Notes in Computer Science*, Springer, pp. 393–402.

[51] LI, L., CHENG, P., OU, L., AND ZHANG, Z. Applying multi-objective evolutionary algorithms to qos-aware web service composition. In *Advanced Data Mining and Applications*. Springer, 2010, pp. 270–281.

[52] LI, L., YANG, P., OU, L., ZHANG, Z., AND CHENG, P. Genetic algorithm-based multi-objective optimisation for qos-aware web services composition. In *Knowledge Science, Engineering and Management*. Springer, 2010, pp. 549–554.

[53] LI, W., AND YAN-XIANG, H. A web service composition algorithm based on global qos optimizing with mocaco. In *Algorithms and Architectures for Parallel Processing, 10th International Conference, ICA3PP*

2010, Busan, Korea, May 21-23, 2010. Proceedings. Part II (2010), C.-H. Hsu, L. T. Yang, J. H. Park, and S.-S. Yeo, Eds., vol. 6082 of *Lecture Notes in Computer Science*, Springer, pp. 218–224.

[54] LI, Y., AND LIN, C. Qos-aware service composition for workflow-based data-intensive applications. In *Web Services (ICWS), 2011 IEEE International Conference on* (2011), pp. 452–459.

[55] MEI XIA, Y., CHEN, J.-L., AND WU MENG, X. On the dynamic ant colony algorithm optimization based on multi-pheromones. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on* (2008), pp. 630–635.

[56] MERZ, P., AND FREISLEBEN, B. Genetic local search for the tsp: New results. In *Evolutionary Computation, 1997., IEEE International Conference on* (1997), IEEE, pp. 159–164.

[57] MILLER, B. L., AND GOLDBERG, D. E. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems 9*, 3 (1995), 193–212.

[58] MING, C., AND ZHEN-WU, W. An approach for web services composition based on qos and discrete particle swarm optimization. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on* (2007), vol. 2, IEEE, pp. 37–41.

[59] MITCHELL, M., HOLLAND, J. H., AND FORREST, S. When will a genetic algorithm outperform hill climbing? In *NIPS* (1993), pp. 51–58.

[60] MURATA, T., AND ISHIBUCHI, H. Performance evaluation of genetic algorithms for flowshop scheduling problems. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on* (1994), IEEE, pp. 812–817.

[61] NANDA, M. G., CHANDRA, S., AND SARKAR, V. Decentralizing execution of composite web services. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications* (2004), OOPSLA '04, ACM, pp. 170–187.

[62] NG, T. S. E., AND ZHANG, H. Towards global network positioning. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement* (New York, NY, USA, 2001), IMW '01, ACM, pp. 25–29.

[63] PERREY, R., AND LYCETT, M. Service-oriented architecture. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on* (2003), IEEE, pp. 116–119.

[64] PHAM, D., AND KARABOGA, D. Intelligent optimisation techniques. *Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Springer, New York* (2000).

[65] POLI, R., LANGDON, W. W. B., AND MCPHEE, N. F. *Field Guide to Genetic Programming*. Lulu Enterprises Uk Limited, 2008.

[66] RUBERG, N., RUBERG, G., AND MANOLESCU, I. Towards cost-based optimization for data-intensive web service computations. In *Proceedings of SBBD, 2004* (2004), p. 283.

[67] RUTENBAR, R. Simulated annealing algorithms: an overview. *Circuits and Devices Magazine, IEEE 5*, 1 (1989), 19–26.

[68] SADIQ, W., SADIQ, S., AND SCHULZ, K. Model driven distribution of collaborative business processes. In *Services Computing, 2006. SCC '06. IEEE International Conference on* (Sept 2006), pp. 281–284.

[69] SINHA, P., AND ZOLTNERS, A. A. The multiple-choice knapsack problem. *Operations Research 27*, 3 (1979), 503–515.

[70] SPEARS, W. M., ET AL. Crossover or mutation. *Foundations of genetic algorithms 2* (1992), 221–237.

[71] SRINIVAS, N., AND DEB, K. Muiltiobjective optimization using non-dominated sorting in genetic algorithms. *Evol. Comput. 2*, 3 (Sept. 1994), 221–248.

[72] STRUNK, A. Qos-aware service composition: A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on* (Dec 2010), pp. 67–74.

[73] TABOADA, H., ESPIRITU, J., AND COIT, D. Moms-ga: A multi-objective multi-state genetic algorithm for system reliability optimization design problems. *Reliability, IEEE Transactions on 57*, 1 (March 2008), 182–191.

[74] TABOADA, H. A., ESPIRITU, J. F., AND COIT, D. W. Moms-ga: A multi-objective multi-state genetic algorithm for system reliability optimization design problems. *Reliability, IEEE Transactions on 57*, 1 (2008), 182–191.

[75] TAN, W., AND FAN, Y. Dynamic workflow model fragmentation for distributed execution. *Computers in Industry 58*, 5 (2007), 381–391.

[76] TAN, W., AND FAN, Y. Dynamic workflow model fragmentation for distributed execution. *Computers in Industry 58*, 5 (2007), 381–391.

[77] THIO, N., AND KARUNASEKERA, S. Automatic measurement of a qos metric for web service recommendation. In *Software Engineering Conference, 2005. Proceedings. 2005 Australian* (March 2005), pp. 202–211.

[78] TSALGATIDOU, A., AND PILIOURA, T. An overview of standards and related technology in web services. *Distrib. Parallel Databases 12*, 2-3 (Sept. 2002), 135–162.

[79] ULDER, N. L., AARTS, E. H., BANDELT, H.-J., VAN LAARHOVEN, P. J., AND PESCH, E. Genetic local search algorithms for the trav-

eling salesman problem. In *Parallel problem solving from nature*. Springer, 1991, pp. 109–116.

[80] WADA, H., CHAMPRASERT, P., SUZUKI, J., AND OBA, K. Multiobjective optimization of sla-aware service composition. In *Services-Part I, 2008. IEEE Congress on* (2008), IEEE, pp. 368–375.

[81] WANG, A., MA, H., AND ZHANG, M. Genetic programming with greedy search for web service composition. In *Database and Expert Systems Applications* (2013), vol. 8056 of *Lecture Notes in Computer Science*, Springer, pp. 9–17.

[82] WANG, J., AND HOU, Y. Optimal web service selection based on multi-objective genetic algorithm. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on* (2008), vol. 1, IEEE, pp. 553–556.

[83] WANG, L., SHEN, J., AND BEYDOUN, G. Enhanced ant colony algorithm for cost-aware data-intensive service provision. In *Services (SERVICES), 2013 IEEE Ninth World Congress on* (2013), pp. 227–234.

[84] WANG, L., SHEN, J., DI, C., LI, Y., AND ZHOU, Q. Towards minimizing cost for composite data-intensive services. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on* (June 2013), pp. 293–298.

[85] WANG, L., SHEN, J., AND YONG, J. A survey on bio-inspired algorithms for web service composition. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* (2012), pp. 569–574.

[86] WODTKE, D., WEISSENFELS, J., WEIKUM, G., AND DITTRICH, A. The mentor project: steps towards enterprise-wide workflow management. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on* (Feb 1996), pp. 556–565.

[87] WU, B., CHI, C.-H., CHEN, Z., GU, M., AND SUN, J. Workflow-based resource allocation to optimize overall performance of composite services. *Future Generation Computer Systems 25*, 3 (2009), 199–212.

[88] YANG, Z., SHANG, C., LIU, Q., AND ZHAO, C. A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. *Journal of Computational Information Systems 6*, 8 (2010), 2617–2622.

[89] YAO, Y., AND CHEN, H. Qos-aware service composition using nsga-ii 1. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (2009), ACM, pp. 358–363.

[90] YILDIZ, U., AND GODART, C. Information flow control with decentralized service compositions. In *Web Services, 2007. ICWS 2007. IEEE International Conference on* (July 2007), pp. 9–17.

[91] YU, T., AND LIN, K.-J. Service selection algorithms for composing complex services with multiple qos constraints. In *Proceedings of the Third International Conference on Service-Oriented Computing*, ICSOC'05. Springer-Verlag, 2005, pp. 130–143.

[92] YU, T., ZHANG, Y., AND LIN, K.-J. Efficient algorithms for web services selection with end-to-end qos constraints. *TWEB 1*, 1 (2007).

[93] YU, Y., MA, H., AND ZHANG, M. A hybrid gp-tabu approach to qos-aware data intensive web service composition. In *Simulated Evolution and Learning* (2014), vol. 8886 of *Lecture Notes in Computer Science*, Springer International Publishing, pp. 106–118.

[94] YU, Y., MA, H., AND ZHANG, M. F-mogp: A novel many-objective evolutionary approach to qos-aware data intensive web

service composition. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on* (May 2015), p. To be published.

[95] YU Y., M. H., AND M., Z. An adaptive genetic programming approach for qos-aware web service composition. In *IEEE Congress on Evolutionary Computation (IEEE CEC 2013)* (2013), IEEE.

[96] YULU, S., AND XI, C. A survey on qos-aware web service composition. In *Multimedia Information Networking and Security (MINES), 2011 Third International Conference on* (Nov 2011), pp. 283–287.

[97] ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., AND SHENG, Q. Z. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), WWW '03, pp. 411–421.

[98] ZENG, L., BENATALLAH, B., NGU, A., DUMAS, M., KALAGNANAM, J., AND CHANG, H. QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng. 30*, 5 (May 2004), 311–327.

[99] ZHANG, L.-J., AND LI, B. Requirements driven dynamic services composition for web services and grid solutions. *Journal of Grid Computing 2* (2004), 121–140.

[100] ZHANG, Q., AND LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on 11*, 6 (Dec 2007), 712–731.

[101] ZHANG, Y., ZHOU, X., AND GAO, Y. Optimizing the data intensive mediator-based web services composition. In *Proc. of the 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development* (2006), APWeb'06, Springer-Verlag, pp. 708–713.

[102] ZHENG, H., ZHAO, W., YANG, J., AND BOUGUETTAYA, A. Qos analysis for web service compositions with complex structures. *Services Computing, IEEE Transactions on 6*, 3 (July 2013), 373–386.

[103] ZITZLER, E., LAUMANNS, M., AND THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. Tech. rep., 2001.

[104] ZITZLER, E., LAUMANNS, M., THIELE, L., ZITZLER, E., ZITZLER, E., THIELE, L., AND THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm, 2001.

[105] ZITZLER, E., AND THIELE, L. *An evolutionary algorithm for multiobjective optimization: The strength pareto approach*. Citeseer, 1998.

[106] ZITZLER, E., AND THIELE, L. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on 3*, 4 (1999), 257–271.