Swarm and Evolutionary Computation 10 (2013) 31–40

Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

Memetic algorithms applied to the optimization of workflow compositions

Simone A. Ludwig

Department of Computer Science, North Dakota State University, Fargo, ND, USA

ARTICLE INFO

Article history:
Received 1 April 2012
Received in revised form
29 October 2012
Accepted 9 December 2012
Available online 11 January 2013

Keywords:
Genetic algorithms
Particle swarm optimization
Munkres algorithm
Quality of service
Memetic algorithms

ABSTRACT

The selection of services of a workflow based on Quality of Service (QoS) attributes is an important issue in service-oriented systems. QoS attributes allow for a better selection process based on non-functional quality criteria such as reliability, availability, and response time. Past research has mostly addressed this problem with optimal methods such as linear programming approaches. Given the nature of service-oriented systems where large numbers of services are available with different QoS values, optimal methods are not suitable and therefore, approximate techniques are necessary. In this paper, we investigate Genetic algorithms and particle swarm optimization for the service selection process. In particular, both methods are combined with an optimal assignment algorithm (Munkres algorithm) in order to achieve higher solution qualities (success ratios) and to form a so called memetic algorithm. Experiments are conducted to investigate the suitability of the approaches and to compare the memetic algorithms with their non-memetic counterparts. The results reveal that the memetic algorithms are very suitable for the application to the workflow selection problem.

1. Introduction

The distribution of software systems has increased over the last decade. There are many different reasons for this increase, and one of them is the need to integrate and connect heterogeneous applications and resources within organizational, but also across organizational boundaries. In particular, most legacy systems and applications were designed for their specific purpose, but not with the view in mind that they need to be integrated with and adapted to different application scenarios. This lack in the design of these legacy systems require new paradigms and approaches to be integrated to cope with these challenges.

Service-orientation provides the conceptual principle necessary to deal with the integration challenges as well as with the increasing complexity by providing adaptive software units referred to as services. Services are characterized by properties such as loose coupling, well-defined service contracts, and standardization that allows them to be independent of any particular implementation technology [1].

The idea behind service-oriented computing is that businesses offer their application functionality as services over the Internet, and users or companies can make use of these services by composing and integrating these services into their applications. Service-oriented architecture is the concept that combines this idea. Fig. 1 describes the basic components of the service-oriented architecture that are the providers, requesters and registries. The service provider publishes the service description in a service registry, and a service requester can query a service from the registry, and dynamically bind it to one of the services that are returned by the search query.

The main idea of service-orientation is to compose these services by discovering them and then dynamically invoking them when building applications, rather than building them from scratch or reusing other applications. Service composition (also called orchestration) enables the development of building service-oriented applications using existing services. The result of this composition process is referred to as a composite service. In this paper, we assume that all available services are validated pre-runtime, so that failure cannot occur during the composition process due to incompatibility. The standard orchestration language is WS-BPEL (Web Service Business Process Execution Language) [2]. Choreography is another term used in service-oriented environments that describes the message interchanges between the different participants in service-oriented systems. It provides the global distributed model of message exchanges without the need of a central coordinator [3]. The Web Service Choreography Description Language (WS-CDL) [4] is one of the first languages for describing the global model of service interactions.

One primary requirement of service-oriented systems is the ability of self-adaptivity [5]. Self-adaptivity in the area of service-oriented environments means that the system should be able to adapt its behavior depending on the changes within its environment. A possible solution for this adaptability, in particular for service composition, is the concept of Quality of Service

E-mail address: simone.ludwig@ndsu.edu

2210-6502/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2012.12.001

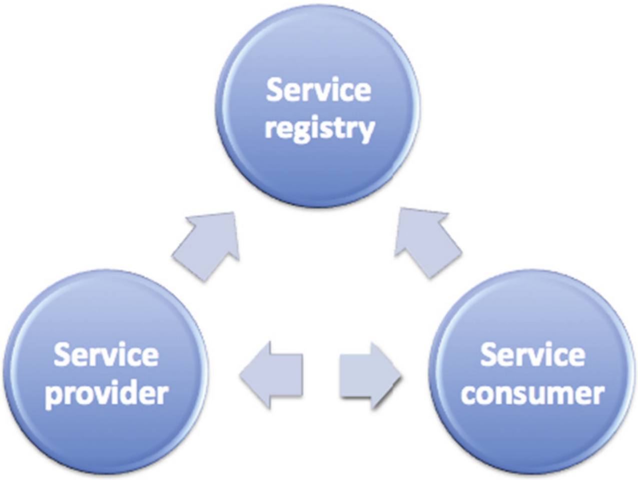

Fig. 1. Service-oriented architecture.

(QoS). All non-functional attributes of a service, such as performance-specific attributes, are described by QoS. QoS attributes can be categorized into deterministic and non-deterministic attributes [6]. Non-deterministic QoS attributes, such as response time, have uncertain values during service invocation. It is necessary to provide as accurate as possible values for all QoS attributes for composite applications and their execution.

A Service Level Agreement (SLA) is a contract between a service provider and a service consumer that captures the agreed-upon terms with respect to QoS parameters. Considering a service-oriented computing environment, capabilities are shared via the implementation of web services exposed to by a service provider. When a service requester requires a specific functionality, which cannot be provided by one single service, the composition of multiple services needs to be done thereby creating a workflow. The composition of web services should not only be functionally compatible, but should also be compatible with regards to the defined service levels. In particular, QoS attributes need to be considered for the dynamic binding process of the concrete services available. Therefore, QoS-aware composition is necessary given the changing and dynamic environment of service-oriented systems (services come online or go offline, new services become available, or existing services change their characteristics).

This paper addresses the workflow selection using approximate algorithms such as Genetic algorithms (GA) and Particle Swarm Optimization (PSO) for the optimization process. Furthermore, the selection of services is based on QoS parameters as well as on service level agreements. Preliminary results have shown that the approximate algorithms achieved an optimized service selection that were higher than that of a random selection and took less time than the optimal algorithm, however, the solution quality needs to be further improved. In order to address this, we apply the memetic algorithm idea of combining an evolutionary algorithm or swarm intelligence algorithm with a local search technique in order to balance the exploration and exploitation of the search space, and therefore, achieving higher solution quality. GA as well as a PSO approach is combined with an optimal search method called the Munkres algorithm. Both algorithms are implemented and experiments show that the combination achieves much higher solution qualities applied to the workflow selection problem than the basic GA and PSO. In particular, it balances the high computational complexity of the Munkres algorithm is balanced with the stochastic advantages of the GA and PSO.

This paper is structured as follows. Section 2 describes related work in the area. In Section 3, background information on workflows, quality of service, and service level agreements are given. Section 4 outlines and describes the approaches implemented and the experiments conducted. The results are displayed in Section 5. And last but not least, Section 6 summarizes the findings and gives an account to future work.

2. Related work

Related work in the area of service and workflow selection can broadly be classified into the following categories: web service infrastructure, agent-based, fuzzy-based, trust-based and optimization approaches. Some but not all of these approaches use QoS as a measure for the selection process.

A dynamic web service selection and composition approach is described in [7]. The approach determines a subset of web services to be invoked during runtime in order to orchestrate a composite web service successfully. A finite state machine model is used to describe the permitted invocation sequences of the web service operations, and a reliability measure is aggregated for the web service operations.

A transactional and QoS-aware selection algorithm is proposed in [8]. The composition of services is recursively constructed based on diverse functionalities, transactional properties and QoS thereby considering the user's requirements. This approach addresses the composition of a workflow based on transactional properties and QoS characteristics using proofs as well as an experimental analysis.

An adaptive hybrid semantic matchmaker for services is proposed in [9]. The matchmaker determines three kinds of semantic service similarity that are logic-based, text-based and structural-based. The degree of structural similarity is computed with the help of the SWDL-analyzer tool by means of XMLS tree edit distance measurement, string-based and lexical comparison of the respective services.

An open, fair, dynamic and secure framework to evaluate the QoS of services is outlined in [6]. The fair computation and enforcement of QoS of web services should have minimal overhead, but yet should be able to achieve sufficient trust by both service requesters and providers. A case study of a phone service provisioning market place application shows the idea of the approach.

The area of agent-based service selection has looked at ways to formulate the problem of service selection. In [10], inspiration from traditional recommender approaches is taken and a new agent-based approach in which agents cooperate to evaluate service providers is proposed. The agents rate each other, and decide on the weight to place on each other's recommendations. The algorithm is devised to work in the context of a concept lattice that enables to find relevant agents.

Another approach [11] developed a multi-agent framework based on an ontology for QoS and a new model of trust. The ontology provides the basis for the providers to advertise their service capabilities and for consumers to express their preference in order for ratings of services to be gathered and shared. The ratings give an empirical measure for the selection of services, and the ratings are quality-specific and obtained by automatic monitoring or user input.

A trustworthy service selection and composition framework based on Bayesian networks and a beta-mixture model is presented in [12]. This approach was devised since existing approaches either failed to capture the dynamic relationships between services or assumed that the environment is fully observable. Experimental results show that their approach punish and reward services in terms of the quality criteria they offer, also this approach is effective even though having to deal with incomplete observations.

Another trust and reputation management approach was introduced in [13]. Their approach tackles the issue to detect and deal with false ratings by dishonest providers and users. A new QoS-based selection approach and ranking solution is given with a formal description, and in addition, experiments are conducted to validate and demonstrate that the solution yields high-quality results under various realistic deception behaviors.

In [14], a system for supporting the user in the discovery of semantic web services is used to model an ad-hoc service request by selecting conceptual terms rather than using strict syntax formats. The selection exploits the fuzzy formal concept analysis to request the system to return a list of semantic web services that match the user query.

A service selection method based on the Technique for Order Preference by Similarity to an ideal Solution (TOPSIS) with fuzzy opinions is used to evaluate the weights of various criteria and the rating of each alternative web service in [15]. The approach uses triangular fuzzy membership functions to represent the weights of criteria and the ratings of web services.

In [16], a fuzzy-based UDDI (Universal Description, Discovery and Integration) with QoS support is proposed to consider the non-functional quality of QoS information for personalized web service selection. This approach considers the objective factors described by service providers, and subjective information with trustability evaluations from users by adapting GA to learn the user preferences, and to apply fuzzy logic to make decisions. The users can determine the most suitable web service with a fuzzy query interface to provide subjective and objective factors.

A decision model under consumer's vague perception of intuitionistic fuzzy set for QoS-aware web services selection is proposed in [17]. The selection method is modeled as a fuzzy multi-criteria decision-making problem by considering the non-functional QoS properties that heavily rely on the perceptions of service providers and consumers.

The workflow service selection problem has received a lot of attention in the past years whereby many linear-programming approaches have been used [18,19]. In [20], complex workflow patterns are used to address the service selection problem, and linear programming is used to solve the optimization problem using an aggregation function for different QoS attributes.

In [21], a quality-driven web service composition approach is outlined applying linear programming. A global planning approach is employed to optimally select component services during the execution of a composite service. However, given that it is an optimal method it does not scale very well with growing search spaces [22].

Some approximate algorithms have also been applied. For example, in [23], a GA approach is used for three QoS parameters (response time, cost, reputation). Basic GA-parameters are varied such as mutation rate, number of generations, fitness function, penalty factor, as well as a comparison of the GA-approach to other heuristics is done. The study reveals that the GA offers a good overall performance, however, not as good when compared to the other heuristics such as branch-and-bound and exhaustive search. This is surprising since GA usually reaches very close to the optimal solution. Their work, however, does not address SLA.

In [22], another GA approach is discussed. They are using four QoS attributes (cost, response time, availability, reliability) and include a factor measuring the ratio of the generations, and the maximum generations as well in their fitness function. Similar to the approach above, no SLA is addressed. Their empirical study compares the GA approach with linear programming. They point out that the linear programming approach does not handle non-linear functions, which the GA method can. Another major point they make is that the GA scales well when the number of services increases compared to the linear programming approach.

A multi-objective optimization based particle swarm optimization algorithm is presented in [24]. The approach solves the global optimization problem for service selection of web services compositions. In particular, it uses a multi-objective constrained optimization with constraints producing a set of constraints to meet the Pareto optimal solution. The QoS parameters considered were execution time, cost, availability, and reliability.

A PSO-based heuristic to schedule applications to cloud resources that takes into account both computation cost and data transmission cost is presented in [25]. Workflow applications by varying its computation and communication costs are compared. The cost savings when using PSO as compared to using existing BRS (Best Resource Selection) algorithm is analyzed. The results show that three times the cost savings can be achieved, as well as a good distribution of workload onto resources is obtained.

Related work done in the past by the author has followed two directions. First, GA and PSO applied to the single service selection problem was investigated [26]. It was discovered that GA and PSO are suitable for the single service assignment since every service request is optimized and a very good service with good QoS parameters is assigned. The second direction addressed the single- vs. multi-objective GA approach [27]. In some studies done by other researchers, the single- vs. multi-objective GA was investigated separately, however, no comparison had been done between the two, and furthermore, no consideration for service level agreements had been incorporated. The findings of this investigation revealed that the single-objective GA approach is better suited by achieving equivalent assignments needing less time for the optimization than the multi-objective approach.

However, compared to an optimal method, the Munkres algorithm [28], the approximate methods (GA and PSO) do not achieve close solution qualities within a reasonable amount of time. In order to address this issue and to improve the solution quality of the approximate methods further, both GA and PSO are combined with the Munkres algorithm. This allows to generate higher solution quality at the same time keeping the execution time of the algorithms within a reasonable range given that the speed of the optimization is paramount in service-oriented environments.

3. Workflow composition

3.1. Workflow example

Fig. 2 shows the abstract as well as the concrete services. In order to illustrate the workflow composition, an example workflow of knowledge discovery in databases, also known as the KDD process [29], is introduced. The process is as follows: the data is first cleaned and preprocessed to remove noise or outlier in the data (AS_1). Then, the data is reduced and projected that includes finding useful features to represent the data (AS_2). Afterwards, the function of the data mining operation has to be chosen (e.g., summarization, classification, regression, and clustering) (AS_3). Then the chosen data mining algorithm is run in search of patterns of interest to be represented in a particular form such as classification rules or trees, regression, clustering, sequence modeling, etc. (AS_4). Once this is done, the results need to be interpreted, and redundant or irrelevant patterns need to be removed, and useful patterns need to be translated into terms understandable by users (AS_5).

The concrete services (CS_{xy}) of each abstract service (AS_x) provide the same functional, but different non-functional properties, i.e., QoS attributes. Selecting the services of a workflow based

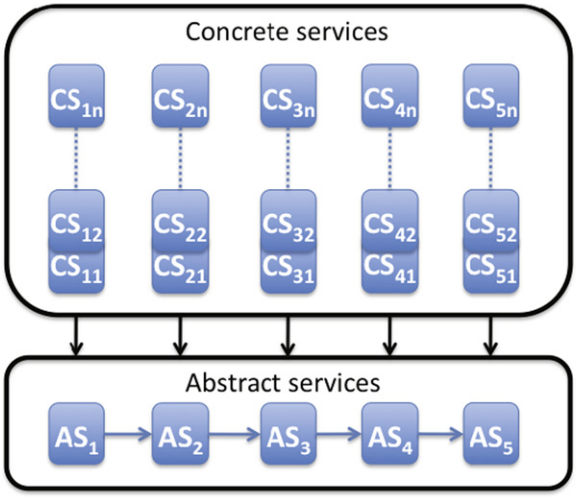

Fig. 2. Example of a knowledge discovery workflow.

on QoS parameters requires an algorithm that can optimize the assignment of concrete services within a workflow for a given abstract workflow description. Furthermore, we are considering that multiple requests are being served at the same time. Given that performance in a service-oriented environment is of essence, we argue that we do not need to have optimal assignments, but close to optimal assignments should be found within a reasonable time.

3.2. *Quality of service metric and objective function*

There are many measures available for different QoS criteria, however, we consider the following four generic quality criteria for single services, also referred to as QoS parameters: reliability, availability, response time, and cost. The first two QoS parameters are to be maximized, whereas the last two are to be minimized.

The reliability $q_1(s)$ of a service is the fraction of requests correctly responded to within a maximum expected time frame. Reliability is a measure related to the hardware and software configuration of web services and their network connections. Reliability values are computed from past data measuring the successful executions in relation to the overall number of executions.

The availability $q_2(s)$ of a service is the fraction of time that the service is accessible. It measures the total amount of time in which the service is available during the last defined period of time (threshold is set by administrator).

The response time $q_3(s)$ denotes the expected delay in seconds from the moment a request is made until the moment when the results are returned. Services advertise their processing time or provide methods to inquire about it.

The execution cost $q_4(s)$ represents the amount of money a user has to pay for executing a service. Web service providers usually advertise the execution price directly or they provide methods to inquire about it. The QoS vector $q(s)$ of a service s is defined as follows: $q(s)=(q_1(s),q_2(s),q_3(s),q_4(s))$.

However, in this study we are concerned not only with single services, but with complete workflows, and therefore, the QoS parameters of the single services have to be aggregated. We assume in our study that we are only using sequential workflows. Therefore, the availability $Q_1(w)$ and reliability $Q_2(w)$ of a workflow w are calculated as the product of each single services availability and reliability respectively (as proposed in [30]). The response time $Q_3(w)$ and execution price $Q_4(w)$ of a workflow w are the sum of each single service's response time and service's cost, respectively.

Therefore, the QoS vector $Q(w)$ of a workflow w is denoted as $Q(w)=(Q_1(w),Q_2(w),Q_3(w),Q_4(w))$.

Our goal is to maximize the selection of services within a workflow based on the QoS parameters. In addition, we are maximizing N workflows at the same time. The single objective function for reliability and availability is

$$f_{obj}=\frac{Q_{ij}-Q_j^{min}}{Q_j^{max}-Q_j^{min}} \tag{1}$$

and the single objective function for response time and cost is

$$f_{obj}=\frac{Q_j^{max}-Q_{ij}}{Q_j^{max}-Q_j^{min}} \tag{2}$$

where by we define Q_{ij} to be the value for workflow i and the jth QoS parameter, and we define Q_j^{max} to be the maximum score any of the considered services achieves for the jth QoS parameter as defined above, i.e., $Q_j^{max}=\max_{s\in S}q_j(s)$ where S is the set of all possible services. And similarly, Q_j^{min} to be the minimum score any of the considered services achieves for the jth QoS parameter ($Q_j^{min}=min_{s\in S}q_j(s)$). Given that the different service levels need to be taken into account we have to define the following constraints:

$$Q_{ij}\geq Q_j(p) \quad \text{for} \quad j=1,2 \tag{3}$$

and

$$Q_{ij}\leq Q_j(p) \quad \text{for} \quad j=3,4 \tag{4}$$

whereby $Q_j(p)$ is the jth QoS value given the chosen service level plan p. The overall objective function for the optimization of the workflows is the following:

$$f_{obj}=\max\sum_{i=1}^{N}\left(\sum_{j=1}^{2}\omega_j\frac{Q_{ij}-Q_j^{min}}{Q_j^{max}-Q_j^{min}}+\sum_{j=3}^{4}\omega_j\frac{Q_j^{max}-Q_{ij}}{Q_j^{max}-Q_j^{min}}\right) \tag{5}$$

Note that the individual QoS parameters are treated differently depending on whether its value is minimized or maximized. Normalized scores are used and each QoS parameter can be weighted differently by parameter w_j.

3.3. *Service level plan*

A service level agreement (SLA) is a contract between a service provider and a service consumer, which captures the agreed-upon terms with respect to QoS parameters. Considering a service-oriented environment, capabilities are shared via the implementation of web services exposed by a service provider. When a service requester requires a specific functionality that cannot be provided by one single service, the composition of multiple services needs to be done thereby creating a workflow. In addition, the composition of web services should not only be functionally compatible, but also should be compatible with regards to the defined service levels. We define the SLA for each user category's reliability, availability, response time, and cost accordingly.

Table 1 outlines the different service level plans (SLP) with normalized service level agreement values that are available to the users. Three different user categories are defined as Platinum, Silver and Gold.

Table 1
Normalized service levels for different service plans.

Service plan	Reliability	Availability	Response time	Cost
Platinum	0.6	0.5	0.7	0.5
Silver	0.7	0.7	0.5	0.65
Gold	0.8	0.9	0.3	0.8

The measure that is observed is the success ratio that measures the percentage of successful workflow compositions in terms of fulfilled SLA (as given in Eqs. (3)–(5)). For example, a success ratio of 90% implies that 90 out of 100 workflow requests fulfill the SLA requirements.

4. Approaches

Since evolutionary algorithms are not best suited for fine-tuning the search in complex combinatorial spaces, researchers have developed hybridization methods to improve the efficiency of the search [31]. Memetic algorithm is the hybridization of using an evolutionary algorithm in combination with a local search technique. Memetic algorithm is seen as extensions of evolutionary algorithms that apply a separate refinement process to improve the solution quality with methods such as hill-climbing or simulated annealing. Memetic algorithms are also known as hybrid evolutionary algorithms [32], Baldwinian evolutionary algorithms [33], Lamarckian evolutionary algorithms [34], cultural algorithms or genetic local search.

The underlying idea of all these algorithms is that they combine evolutionary algorithm operators with local search heuristics. Furthermore, combinations with constructive heuristics or optimal methods may also be considered within this category of algorithms (and this is what we are making use of). Memetic algorithms have been shown to be more efficient and more effective than the evolutionary algorithms alone for certain problem domains. Memetic algorithms can achieve higher solution quality in fewer numbers of generations. In particular, for many combinatorial optimization problems memetic algorithms have proven themselves to be very effective. An example of such is the quadratic assignment problem and the traveling salesman problem [35].

For more information on memetic algorithms, the reader is referred to an extensive review that is given in [36].

4.1. Munkres algorithm

The Munkres Algorithm is an optimal combinatorial optimization algorithm, and was first known as the Hungarian algorithm. The Hungarian algorithm received its name by Kuhn in 1955 [37,38] since it was based on the earlier works of two Hungarian mathematicians (Denes Koening and Jeno Egervary). Two years later, Munkres reviewed and enhanced the algorithm and has been known as the Munkres algorithm [28,39] ever since. The original algorithm was $O(n^4)$, however, it was modified to achieve an improved complexity of $O(n^3)$.

The assignment problem as formally defined by Munkres in 1957 [38] is "Let r_{ij} be a performance ratings for a man M_i for job J_j. A set of elements of a matrix are said to be independent if no two of them lie in the same line (the word "line" applies both to the rows and to the column of a matrix). One wishes to choose a set of n independent elements of the matrix (r_{ij}) so that the sum of the element is minimum."

Similarly, the problem of workflow selection can be defined as an $n \times m$ requester-workflow matrix, representing the success ratio of each requester with every workflow combination. The Munkres algorithm works on this matrix, to assign the requests to workflows as to achieve an overall optimal success ratio.

A detailed description of the steps of the Munkres algorithm is given as follows:

1. **Step 1**: A $n \times m$ matrix is created, called the success ratio matrix, in which each element represents the success ratio of assigning one of n requests to one of m workflows.
2. **Step 2**: For each row of this matrix, the highest value of the success ratio is found, and is subtracted from every element in the row. The absolute values are taken.
3. **Step 3**: A zero (Z) is searched for in the resulting matrix. If there is no starred zero in the row or column, the Zero is starred Z. This is repeated for each row in the matrix.
4. **Step 4**: Then each column containing a starred zero is covered. If K columns are covered, the starred zeros describe a complete set of unique assignments. If this is the case, the algorithm continues with Step 8, otherwise, with Step 5.
5. **Step 5**: A non-covered zero in the matrix is found, and is primed. If there is no starred zero in the row containing a primed zero, the algorithm continues with Step 6. Otherwise, this row is covered and the column containing the starred zero is uncovered. This continues until there are no uncovered zeros left. Finally, the smallest uncovered value is saved and the algorithm continues with Step 7.
6. **Step 6**: A series of alternating primed and starred zeros are constructed in this step. Let Z_0 represent the uncovered primed zero found in Step 5. Let Z_1 denote the starred zero in the column of Z_0. Let Z_2 denote the primed zero in the row of Z_1. This series of alternating primed and starred zero construction is continued until the series terminates at a primed zero that has no starred zero in its column. Then, each of the starred zero of the series is unstarred, and each of the primed zero of the series is starred, and finally all the primes are erased and every line in the matrix is uncovered; and the algorithm continues with Step 4.
7. **Step 7**: The value found in Step 5 is added to every element of each covered row, and is subtracted from every element of each uncovered column. The algorithm loops back to Step 5 without altering any stars, primes, or covered lines.
8. **Step 8**: The assignment pairs are indicated by the positions of the starred zeros in the success ratio matrix. If *success ratio*(i,j) is a starred zero, then the element associated with row i is assigned to the element associated with column j, is added and then divided by the total number of successes to obtain the overall success ratio.

4.2. Memetic algorithm (MA) approach

A GA is a heuristic used to find approximate solutions to difficult-to-solve problems by applying the principles of evolutionary biology such as biologically derived techniques of inheritance, mutation, natural selection, and recombination (or crossover) [40]. GAs are implemented as an algorithm in which a population of solutions (or individuals) to an optimization problem evolve towards better solutions. This is enabled since each solution is a chromosome that can undergo genetic modification.

The workflow selection problem has been encoded with the following chromosome representation:

SLP	NoS	CS1	CS2	CS3	SLP	NoS	CS1	CS2	CS3	CS4	...
2	3	13	26	31	1	4	18	23	37	42	...

A gene within the chromosome consists of integers, whereby the first number characterizes the SLP, the second characterizes the number of services (NoS) that the workflow consists of, and the following integers are the concrete services of the workflow. The first digit from the left of a concrete service (CS) characterizes the abstract service number; the second describes the specific concrete service implementation. The number of services the abstract workflow consists of determines the length of the gene. For example, the first workflow consists of three services, the second of four services, etc. The service level guides the fitness calculations as described in the previous subsection, i.e., the fitness value is determined depending on the service level. For this study, we consider workflows up to five services, having 10 concrete services available for each of the 10 abstract services.

The algorithmic description of the MA approach is given in Algorithm 1. The process of the optimization starts with a population of completely randomly generated individuals. In each generation, the fitness of each population member is evaluated. The fittest individuals, in terms of best fitness value, e.g. from an archive population, where the best solutions found so far are saved. As even the quality of solutions can range widely, particularly in earlier generations, members compete in tournaments, with winners forming a mating pool. Two parents are randomly selected from the pool, and undergo cycle crossover [41] and mutation to form two children. This is repeated until the new population of size N is filled. Then, for a given number of the population size (e.g., 10%), the individuals are randomly chosen to perform the Munkres algorithm on this partial selection. If after applying the Munkres algorithm an improvement is achieved, then the individuals are updated with the optimized selected services and the next generation continues until the stopping criteria is met.

Configurable parameters in the implementation include number of generations as termination criterion, tournament size (the size of the tournament used to select parents), crossover probability, effected positions (how many positions are set to crossover in the crossover mask), and mutation probability.

Algorithm 1. MA algorithm.

```
Input: Munkres portion n
initialize random population $P$
**repeat**
  **for** $i=1$ to $P$ **do**
    select parents from $P$
    generate offspring applying recombination to parents selected
    evaluate fitness
  **end for**
  run Munkres algorithm on n% of population and replace chromosomes with higher fitness
**until** stopping criterion is satisfied
```

4.3. Memetic particle swarm optimization (MPSO) approach

PSO, as introduced in [42], is a swarm based global optimization algorithm. It models the behavior of bird swarms searching for an optimal food source. The movement of a single particle is influenced by its last movement, its knowledge, and the swarm's knowledge. In terms of a bird swarm this means, a bird's next movement is influenced by its current movement, the best food source it ever visited, and the best food source any bird in the swarm has ever visited.

PSOs basic equations are

$$x_i(t+1)=x_i(t)+v_{ij}(t+1) \tag{6}$$

$$v_{ij}(t+1)=w(t)v_{ij}(t)+c_1r_1(t)(xBest_{ij}(t)-x_{ij}(t))+c_2r_{2j}(t)(xGBest_j)-x_{ij}(t)) \tag{7}$$

where x represents a particle, i denotes the particle's number, j the dimension, t a point in time, and v is the particle's velocity. *xBest* is the best location the particle ever visited (the particle's knowledge), and *xGBest* is the best location any particle in the swarm ever visited (the swarm's knowledge). w is the inertia weight and used to weigh the last velocity, c_1 is a variable to weigh the particle's knowledge, and c_2 is a variable to weigh the swarm's knowledge. r_1 and r_2 are uniformly distributed random numbers between zero and one.

PSO is commonly used on real and not discrete problems. In order to solve the discrete workflow selection problem using the PSO approach, several operations and entities have to be defined. This implementation follows the implementation for solving the traveling salesman problem as described in [31].

First, a swarm of particles is initialized. A single particle represents a possible workflow, i.e., every particle's position in the search space must correspond to a possible workflow. The workflow selection is implemented as a vector. Velocities are implemented as lists of changes that can be applied to a particle (its vector) and will move the particle to a new position (a new workflow). Further, *minus* between two particles, *multiplication* of a velocity with a real number, and *addition* of velocities have to be defined. *Minus* is implemented as a function of particles. This function returns the velocity containing all changes that have to be applied to move from one particle to another in the search space. *Multiplication* randomly deletes single changes from the velocity vector, if the multiplied real number is smaller than one. If the real number is one, no changes are applied. For a real number larger than one, random changes are added to the velocity vector. When a velocity is *added* to another velocity, the two lists containing the changes will be concatenated.

MPSO is described in Algorithm 2. First of all, the particles are randomly initialized, then each particle is evaluated by applying Eqs. (6) and (7). If the new position is better than the particle's best position, the particle's best value is updated. If the particle's new position is better than the swarm's best position, then the swarm's best position is updated. Each particle's velocity and position are updated, and then the Munkres algorithm is run on a certain number of particles (e.g., 10%). If after applying the Munkres algorithm an improvement is achieved, then the appropriate particles are updated and the next generation continues until the stopping criterion is satisfied.

The PSO implemented uses guaranteed convergence, which means that the best particle is guaranteed to search within a certain radius, implying that the global best particle will not get trapped in a local optima. Configurable parameters in the implementation include numbers of particles (size of the swarm), number of generations, c_1 (the weighting of the local knowledge), c_2 (the weighting of the global knowledge), w (the weighting of the last velocity), radius (defines the radius in which the global best particles searches randomly), global best particle swarm optimization (determines whether global best particle swarm or local best particle swarm optimization is used), and neighborhood size (defines the neighborhood size for local best particle swarm optimization).

Algorithm 2. MPSO algorithm.

```
Input: Munkres portion $n$
Initialize the swarm of particles
**repeat**
  **for** each particle $p$ **do**
    $value_p \leftarrow evaluate(x_p)$
    **if** $value(x_p) < value(pbest_p)$ **then**
      $pbest_p \leftarrow x_p$
    **end if**
    **if** $value(x_p) < value(gbest)$ **then**
      $gbest \leftarrow x_p$
    **end if**
  **end for**
  **for** each particle $p$ **do**
    $velocity_p \leftarrow updateVelocity()$
```

```
        $x_p \leftarrow updatePosition(x_p, velocity_p)$
    end for
    apply Munkres algorithm on n% of particles and replace the
  particles with higher fitness
until stopping criterion is satisfied
```

5. Experiments and results

5.1. Experimental setup

All four algorithms were implemented in Java and experiments were designed to measure the success ratio and the execution time. Measurements include analysis of success ratio and execution time for different numbers of generations, numbers of individuals used, numbers of particles used, different variations of the weights in the fitness function, and the scalability of the approaches. Thirty runs were conducted in order to account for the stochastic nature of the algorithms. The data set for the services and workflows were randomly generated. Workflows were generated consisting up to five abstract services, out of a pool of 10 concrete services for each of the ten abstract services, i.e., 100 concrete services. Note that we assume that a concrete service can be used in several workflows, and no maximum bound is given for simultaneous calls.

The following parameters have been chosen due to their superior performance on preliminary runs of the workflow selection problem, balancing between accuracy and execution time, also with regards to scalability. In addition, the parameters of GA and PSO were set in order for both algorithms to achieve comparable success ratios and execution times.

For GA/MA, the parameters were set to:

- population size = 100
- number of generations = 1000
- crossover probability = 60%
- mutation probability = 0.5%
- size of the tournament selection = 10
- number of positions that are selected for crossover = 10%
- selection of individuals applied on Munkres algorithm = 10%

For PSO/MPSO, the parameters were set to:

- number of particles = 100
- number of generations = 1000
- $w = 0.5$
- $c_1 = 1.8$
- $c_2 = 0.05$
- radius = 5.0
- global best PSO
- selection of particles applied on Munkres algorithm = 10%

The experiments were conducted on an Intel Core 2 Duo (2.4 GHz, 3 MB L2 cache) running the Java version 1.6.2 JDK runtime environment.

5.2. Results

The results are presented in the following order: first, the success ratios and execution times of all algorithms are displayed showing the difference of both measures of the non-memetic (GA and PSO) and the memetic (MA and MPSO) algorithms; then MA and MPSO are further analyzed for different numbers of generations; then MA is analyzed with regards to the numbers of individuals used, and similarly MPSO with regards to the numbers of particles used; then, the scalability of the approaches is investigated; followed by an analysis for varying percentages of Munkres portions.

Table 2 shows the results of the success ratios and execution times for all algorithms after 200 generations. The success ratios for both GA and PSO are relatively low compared to MA and MPSO. Both GA and PSO achieve success ratios of around 87%, whereas MA and MPSO achieve values around 98%. Comparing MA with MPSO shows that the success ratio of MPSO is 98.7%, whereas MA has a success ratio of 98.3% after 200 generations. This clearly demonstrates that both MA and MPSO achieve higher success ratios, however, this comes at a price of increased execution times. GA and PSO run the 200 generations in around 53 s, whereas MA and MPSO need around 89 s. Comparing MA with MPSO reveals that MA has a slightly shorter execution time than PSO; measured are 88.6 s compared to 93.4 s for MPSO.

In order to find out whether the MA and MPSO algorithms really improve the success ratio compared to their base algorithms, GA and PSO were run for the same amount of time and the success ratio was measured. The results are that the GA achieved a success ratio of 90.3% when run for 88.6 s (as compared to MA achieving 98.3%), and PSO achieved a ratio of 90.8% when run for 93.4 s (as compared to MPSO achieving 98.7%). This confirms that the local search method within the memetic variants improve the success ratio by a significant portion.

Fig. 3 shows the success ratios achieved by MA and MPSO for increasing numbers of generations. Both algorithms start with success ratios of around 86% after 10 generations, and achieving almost 100% after 200 generations.

The execution times shown in Fig. 4 display almost a linear trend for 200 generations. As pointed out previously, MA scales slightly better than MPSO needing less time for the optimization process.

Figs. 5 and 6 show the success ratios and execution times for increasing population sizes of MA, respectively. The success ratios range between 93.9% and 94.8% for a population size of 20 and 200, respectively. A linear trend can be seen in Fig. 5 within the range of variation of ±0.04%. As for the execution times, shown in Fig. 6, it increases first linearly for small population sizes, however, around a population size of 150 it shows a non-linear trend.

Table 2
Success ratios and execution times of all algorithms.

Algorithms	Success ratio (%)	Execution time (s)
GA	86.8	53.5
MA	98.3	88.6
PSO	87.3	53.1
MPSO	98.7	93.4

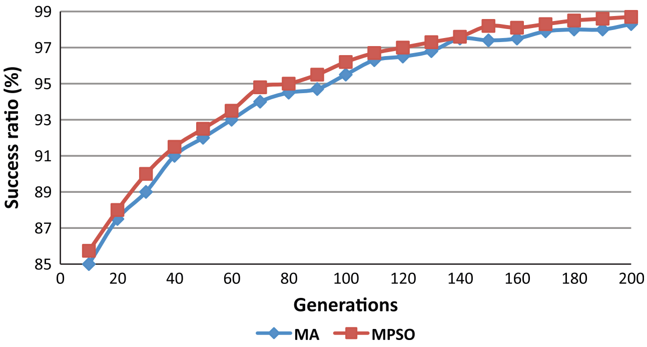

Fig. 3. Success ratios for increasing generations.

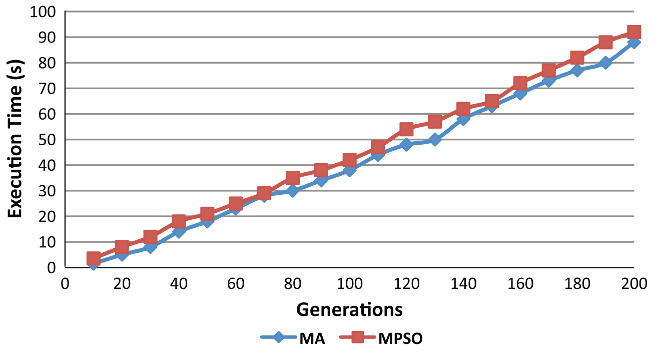

Fig. 4. Execution times for increasing generations.

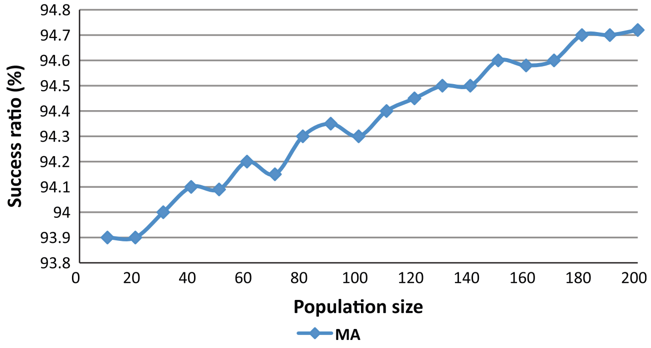

Fig. 5. Success ratios for increasing population sizes.

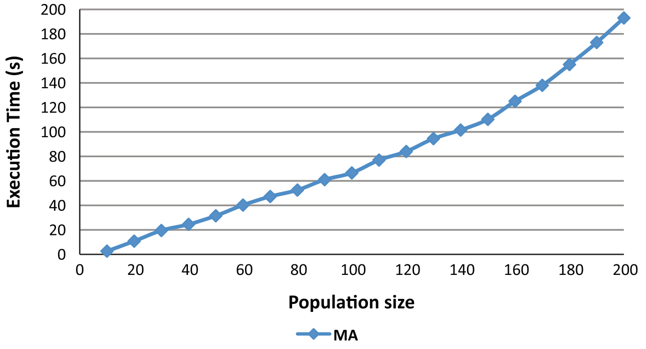

Fig. 6. Execution times for increasing population sizes.

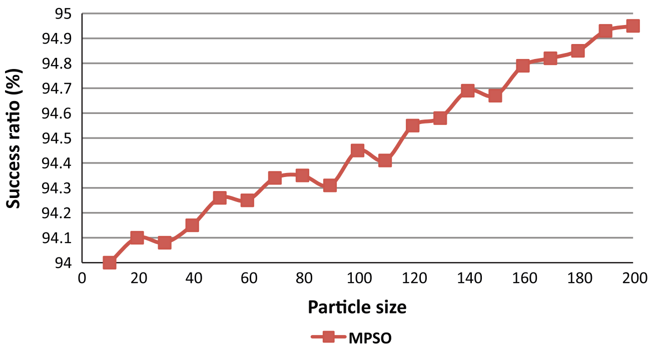

Fig. 7. Success ratios for increasing particle sizes.

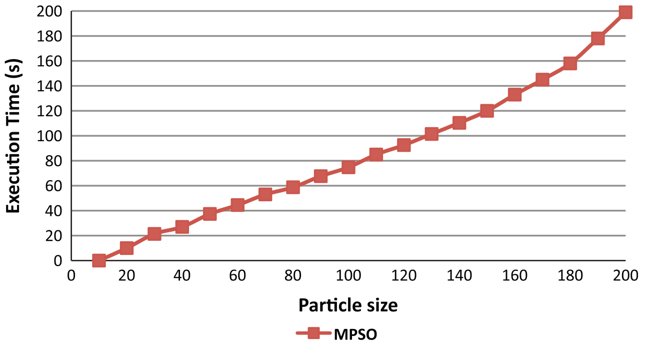

Fig. 8. Execution times for increasing particle sizes.

Table 3
Success ratios and execution times for increasing % of Munkres portion.

Munkres portion (%)	MA		MPSO	
	Success ratio (%)	Exec. time (s)	Success ratio (%)	Exec. time (s)
5	89.4	25.8	90.1	27.6
10	93.6	31.1	94.3	33.8
15	95.2	47.9	96.6	49.8
20	97.9	66.4	98.0	69.7

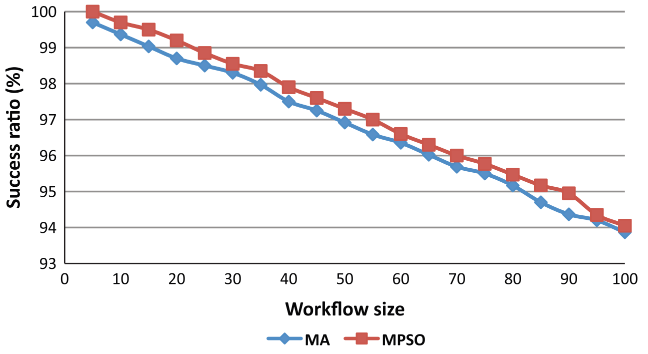

Fig. 9. Success ratios for increasing workflow sizes.

Figs. 7 and 8 show the success ratios and the execution times of varying particle sizes, respectively. The success ratios show a slight linear increase with increasing particle sizes within the range of variation of ±0.03%. The execution times seem to follow a similar trend as the ones for the MA algorithm. For the first numbers of particle sizes we can observe a nearly linear increase, whereas for particle sizes of 160 and higher a non-linear trend is observed.

Table 3 shows the different success ratios and execution times for both algorithms for different Munkres portions ranging from 5% to 20%. In terms of execution times, we can observe the high computational cost the Munkres portion has on both algorithms. As for the success ratio, larger increases can be seen first with smaller increases following. The number of generations chosen was 50.

In order to investigate the scalability of the approach, different workflow sizes ranging from 5 to 100 workflows are analyzed. The number of generations chosen was only 50 in order to show the effect of decreasing success ratios more evidently. Fig. 9 shows how the success ratio decreases from around 100% to 93.4% for both approaches. As seen before, MPSO achieves slightly higher success ratios than MA, e.g., for a workflow size of 50, MPSO scores 96.9%, whereas MA scores 96.7%.

The execution times for the Munkres, MA and MPSO are shown in Fig. 10. Note that the y-axis is in logarithmic scale and displays minutes. It reveals Munkres $O(n^3)$ complexity and demonstrate that the Munkres algorithm cannot be used on the workflow selection problem due to the enormous execution times for larger problem sizes.

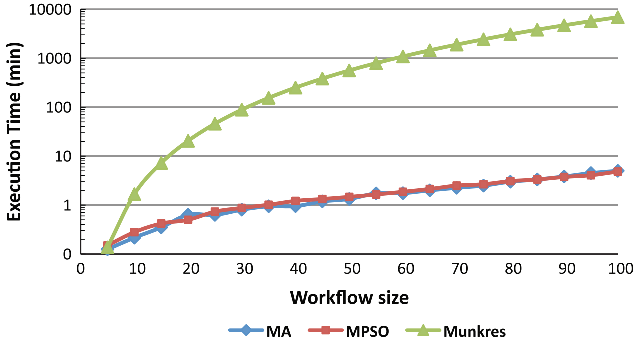

Fig. 10. Execution times for increasing workflow sizes including Munkres algorithm.

6. Conclusions

This paper addressed the service composition task of workflows applying the concept of memetic algorithms to GA and PSO. Memetic algorithms are usually a combination of an evolutionary algorithm with a local search method. In this paper, given that we have a combinatorial optimization problem, GA and PSO have been combined with the Munkres algorithm, which is an optimal combinatorial assignment algorithm.

In the area of service-oriented systems, the service selection process has been done primarily using linear programming methods. However, given that linear methods do not scale well with increasing problem sizes, i.e., workflow sizes, an approximate method is paramount. The approximate methods such as GA and PSO achieve an optimized assignment in a reasonable amount of time. However, in order to further improve the solution quality, and tackling the problems the approximate methods face, namely the balance between exploitation and exploration, we have combined GA and PSO with the Munkres algorithm, therefore, achieving the benefits the memetic algorithms enjoy.

The results show that the memetic algorithms achieve higher success ratios than their non-memetic counterparts. The execution times of the memetic algorithms are as expected higher than the non-memetic algorithms.

Comparing the success ratio of both MA and MPSO showed that MPSO has slightly higher success ratios than MA. Furthermore, analyzing MA for increasing population sizes revealed a slight upward trend towards higher success ratios. Similarly, looking at MPSO a slight improvement in success rations is observed for increasing particle sizes.

The effect of different Munkres portions on both memetic algorithms was investigated. In terms of execution times, the high computational cost the Munkres portion places on both algorithms can be observed when the portion gets larger. As for the success ratio, larger increases are seen first with smaller increases following. This reveals that a good balance between improved success ratio and execution time needs to be chosen.

Given that scalability is a very important issue in service-oriented environments, different workflow sizes were investigated. The success ratio dropped substantially for larger workflow sizes. This shows that if the success ratio needs to remain constant for different numbers of workflow sizes the number of generations needs to be increased, as well as the population size and particle size for MA and MPSO needs to be increased, respectively. However, this comes at the cost of higher execution times, and therefore, need to be traded off carefully.

Overall, a general recommendation is to make use of the memetic algorithms for the workflow selection optimization since they achieve very good success ratios with slightly larger execution times than the non-memetic algorithms, by having an improved execution time as opposed to using an optimal algorithm such as the Munkres algorithm.

Future work will expand this line of research by taking the following constraints imposed by the real world setting of service-oriented systems into consideration. First of all, service invocations of a particular service are limited, and therefore, need to be taken into account. In addition, failure of the service execution and recomposition needs to be addressed, and a solution needs to be implemented and evaluated. Furthermore, since the workflows and concrete services were fairly similar in terms of range, the effects of larger variations on the QoS values of the workflow services need to be investigated.

References

[1] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D.F. Ferguson, Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, 1st ed., Prentice Hall, 2005.

[2] OASIS – Web Service Business Process Execution Language 2.0, 2006. Available from 〈http://www.oasis-open.org/committees/tchome.php/wgabbrev-wsbpel〉 (last retrieved April 2012).

[3] J.M. Zaha, M. Dumas, A.H.M. ter Hofstede, A.P. Barros, G. Decker, Service interaction modeling: bridging global and local views, in: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC06), Hong Kong, China, October 2006.

[4] W3C – Web Services Choreography Description Language (WS-CDL), November 2005. Available from 〈http://www.w3.org/TR/ws-cdl-10/〉 (last retrieved April 2012).

[5] G. Denaro, M. Pezze, D. Tosi, Designing self-adaptive service-oriented applications, in: Proceedings of the Fourth International Conference on Autonomic Computing (ICAC07), Jacksonville, FL, USA, 2007.

[6] Y. Liu, A.H. Ngu, L. Zeng, QoS computation and policing in dynamic web service selection, in: Proceedings of the Thirteenth International Conference on World Wide Web (WWW04), New York, NY, USA, May 2004.

[7] S.-Y. Hwang, E.-P. Lim, C.-H. Lee, C.-H. Chen, Dynamic web service selection for reliable web service composition, IEEE Transactions on Services Computing (2008) 104–116.

[8] M. El Hadad, J. Manouvrier, M. Rukoz, TQoS: transactional and QoS-aware selection algorithm for automatic web service composition, IEEE Transactions on Services Computing 3 (1) (2010) 73–85.

[9] M. Klusch, P. Kapahnke, I. Zinnikus, Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer, in: Proceedings of the Sixth European Semantic Web Conference on The Semantic Web: Research and Applications, 2009.

[10] R. Sreenath, M.P. Singh, Agent-based service selection, Journal of Web Semantics: Science, Services and Agents on the World Wide Web 1 (3) (2011).

[11] E.M. Maximilien, M.P. Singh, Multiagent system for dynamic web services selection, in: Proceedings of the AAMAS Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE), Utrecht, July 2005.

[12] C.-W. Hang, M.P. Singh, Trustworthy service selection and composition, ACM Transactions on Autonomous and Adaptive Systems 6 (February (1)) (2011) 5:1–5:17.

[13] L. Vu, M. Hauswirth, K. Aberer, QoS-based service selection and ranking with trust and reputation management, in: Proceedings of the Cooperative Information System Conference (CoopIS), 2005.

[14] G. Fenza, S. Senatore, Friendly web services selection exploiting fuzzy formal concept analysis, Soft Computing – A Fusion of Foundations, Methodologies and Applications 14 (8) (2010) 811–819.

[15] C. Lo, D. Chen, C. Tsai, K. Chao, Service selection based on fuzzy TOPSIS method, in: 2010 IEEE Twenty-fourth International Conference on Advanced Information Networking and Applications Workshops, 2010, pp. 367–372.

[16] H. Wang, C. Lee, T. Ho, Combining subjective and objective QoS factors for personalized web service selection, Expert Systems with Applications 32 (2) (2007) 571–584.

[17] P. Wang, QoS-aware web services selection with intuitionistic fuzzy set under consumers vague perception, Expert Systems with Applications 36 (3) (2009) 4460–4466.

[18] T. Yu, Y. Zhang, K.J. Lin, Efficient algorithms for web services selection with end-to-end QoS constraints, ACM Transactions on the Web 1 (1) (2007).

[19] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Transactions on Software Engineering 33 (6) (2007) 369–384.

[20] D. Schuller, J. Eckert, A. Miede, S. Schulte, R. Steinmetz, QoS-aware service composition for complex workflows, in: Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services, 2010.

[21] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q.Z. Sheng, Quality driven web services composition, in: Proceedings of the Twelfth International Conference on World Wide Web, 2003.

[22] G. Canfora, M.D. Penta, R. Esposito, M.L. Villani, An approach for QoS-aware service composition based on genetic algorithms, in: Proceedings of Conference on Genetic and Evolutionary Computation, 2005.

[23] M.C. Jaeger, G. Muehl, QoS-based selection of services: the implementation of a genetic algorithm, in: Proceedings of Conference on Communication in Distributed Systems, Workshop on Service-Oriented Architectures and Service-Oriented Computing, 2007.

[24] H. Xia, Y. Chen, Z. Li, H. Gao, Y. Chen, Web service selection algorithm based on particle swarm optimization, in: Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, December 2009, pp. 467–472.

[25] S. Pandey, L. Wu, S.M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10), 2010.

[26] S.A. Ludwig, T. Schoene, Web service selection using particle swarm optimization and genetic algorithms, in: Proceedings of Third World Congress on Nature and Biologically Inspired Computing (NaBIC), Salamanca, Spain, October 2011.

[27] S.A. Ludwig, Single-objective versus multi-objective genetic algorithms for workflow composition based on service level agreements, in: Proceedings of IEEE International Conference on Service Oriented Computing & Applications (SOCA 2011), Irvine, California, USA, December 2011.

[28] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics 5 (32) (1957).

[29] M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery: an overview, in: Advances in Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, Menlo Park, CA, 1996, pp. 1–34.

[30] J. Cardoso, A.P. Sheth, J.A. Miller, J. Arnold, K.J. Kochut, Modeling quality of service for workflows and web service processes, Web Semantics Journal: Science, Services and Agents on the World Wide Web Journal 1 (3) (2004) 281–308.

[31] M. Clerc, Discrete particle swarm optimization – illustrated by the traveling salesman problem, New Optimization Techniques in Engineering, Springer, 2004.

[32] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.

[33] Z. Michalewicz, D.B. Fogel, How to Solve It: Modern Heuristics, Springer-Verlag, 2002.

[34] J.-R. Zhanga, J. Zhanga, T.-M. Lokc, M.R. Lyud, A hybrid particle swarm optimization back-propagation algorithm for feedforward neural network training, Journal of Applied Mathematics and Computation 185 (February (2)) (2007) 1026–1037.

[35] M. Clerc, J. Kennedy, The particle swarm explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.

[36] F. Neri, Memetic algorithms and memetic computing optimization: a literature review, Swarm and Evolutionary Computation 2 (February (1)) (2012) 1–12.

[37] H.W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics 52 (1) (1955).

[38] H.W. Kuhn, The hungarian method for solving the assignment problem, Naval Research Logistics Quarterly 2 (1955) 83.

[39] F. Bourgeois, J.C. Lassalle, An extension of the Munkres algorithm for the assignment problem to rectangular matrices, Communications of the ACM 14 (12) (1971).

[40] M. Mitchell, An Introduction to Genetic Algorithms (Complex Adaptive Systems), The MIT Press, 1998, ISBN 0-262-63185-7.

[41] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, C. Whitley, A comparison of genetic sequencing operators, in: Rick Belew, Lashon Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, San Mateo, CA, 1991, pp. 69–76.

[42] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995.