Abstract—In services computing, service discovery is an important activity to search the pertinent service to the client's requirement. With increasing keywords the search space grows above average and brute force is an inappropriate approach. A cloud service provider requires a flexible and an easy to use application. This practical research presented in this paper combines an intelligent solution based on nature-inspired swarm intelligence for keyword-based service matching with an innovative and user-friendly graphical user interface. Finding optimum parameter settings is a time-consuming and difficult task. Objective of the integrated Parameter Recommender is the determination of optimum values for the applied heuristic.

Index Terms—Dynamic service composition, keyword matching, ant colony optimization, parameter recommendation.

I. INTRODUCTION

In the environment of cloud computing, users can access web service easily by SOA (Service-Oriented Architecture). The high interoperability, cross-platform and loose coupling characteristics of web service fully appear. The principles of SOA and realization of web services motivate the organizations and developers using this scalable and flexible technology. In Services Computing, service discovery is an important activity to search the pertinent service to the client’s requirement [1]. In the world of web services, many complex business processes involve multiple steps being performed in sequence to achieve the business goals. The complexity is broken down into manageable elements, in this case searchable web services. These elements are dynamically composed to units of greater structure to fulfill business requirements.

The idea behind the keywords is a standardized description of the characteristics and behaviour of a service offered in a cloud computing environment [2].

II. PROBLEM

With increasing keywords the search space grows above average [2]. Based on the uniform distribution it is assumed, that there is nearly the same number of services for every pair of keywords that these keywords contain, in this example:

\[ M = \{\text{services contain 1}, \{2\}, \{3\}\} \]  

(1)

Without loss of generality it is assumed, that the lengths of the sets are constant and equal. Based on research the value for constant \(c\) is fixed at the length of 100. For the calculation of the optimum solution based on the subsets of \(M\) using the brute force approach, all possible compositions are build in the sequence described below:

1) each service from 1-3 separately
2) for each 2-combination from the subsets \(M\):
   - each service from 1 combined with each in 2
   - each service from 1 combined with each in 3
   - each service from 2 combined with each in 3
3) for each 3-combination from the subsets \(M\):
   - Each service from 1 combined with each service in 2 combined with each service in 3.

The search space for 3 keywords is calculated as follows:

\[ 3 \times 100 \times 100 \times 100 = 300000 \]

\[ 1 \times 100 \times 100 \times 100 = 1000000 \]

\[ 1+2+3 = 1030300 \] different service compositions that are evaluated sequentially by the brute force approach.

The search space, or number of service compositions to be examined for \(n\) keywords is calculated in general with the following formula:

\[ \sum_{i=1}^{n} \binom{n}{i} \times c^i = (c+1)^n - 1 \]  

(2)

where

\(n\) = number of requested keywords

\(c\) = cluster constant

<table>
<thead>
<tr>
<th>number of keywords ((n))</th>
<th>cluster constant ((c))</th>
<th>number of possible combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100</td>
<td>1,030,300</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>104,060,400</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>10,510,100,500</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>1,061,520,150,600</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>107,213,535,210,700</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>10,828,567,056,280,800</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>1,093,685,272,684,360,000</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>110,462,212,541,120,000,000</td>
</tr>
</tbody>
</table>

III. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) is a recently proposed metaheuristic approach for solving hard combinatorial optimization problems [3]. The inspiring source of ACO is the pheromone trail laying and following behaviour of real ants which use pheromones as a communication medium. In analogy to the biological example, ACO is based on the indirect communication of a colony of simple agents, called (artificial) ants, mediated by (artificial) pheromone trails. The pheromone trails in ACO serve as distributed, numerical information which the ants use to probabilistically construct

Manuscript received June 19, 2014; revised September 20, 2014.

The authors are with Baden-Wuerttemberg Cooperative State University, Germany (e-mail: carsten.mueller@itg-research.net).

DOI: 10.7763/LNSE.2015.V3.208
solutions to the problem being solved and which the ants adapt during the algorithm’s execution to reflect their search experience.

Artificial ants used in ACO are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to partial solutions. These procedures take into account 1) heuristic information on the problem instance being solved, if available and 2) (artificial) pheromone trails which change dynamically at run-time to reflect the agents’ acquired search experience [4]. A stochastic component in ACO allows the ants to build a wide variety of different solutions and hence explore a much larger number of solutions than greedy heuristics. At the same time, the use of heuristic information, which is readily available for many problems, can guide the ants towards the most promising solutions.

A colony of ants concurrently and asynchronously moves through adjacent states of the problem by building paths on the graph. They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information. By moving, ants incrementally build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution and deposits pheromone trails on the components or connections it used [5]. This pheromone information will direct the search of the future ants [4]. Besides ants' activity, an ACO algorithm includes two more procedures: pheromone trail evaporation and daemon actions (this last component being optional). Pheromone evaporation is the process by means of which the pheromone trail intensity on the components decreases over time. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm towards a sub-optimal region. It implements a useful form of forgetting, favoring the exploration of new areas of the search space. Daemon actions are used to implement centralized actions which cannot be performed by single ants.

IV. Solution

A. Algorithm Setup

10,000 services and 250 unique keywords are stored in a memory-based repository using state-of-the-art hash algorithms. Each service contains two to five keywords and a value between 250 and 2500 representing the costs of a service. When setting up the repository in which the services are stored, the memory-based map-object map is generated which supports the reduction of the applied search area.

The algorithm starts with the given keywords based on the request from the user. In this example the requested keywords are “4 2 8” which are split into “4”, “2” and “8”. Based on this input a set M is build by looking up the given keywords in the previously generated map.

\[ M := \{ \text{services that contain keyword 1} \} \]
\[ \{ \text{services that contain keyword 2} \} \]
\[ \{ \text{services that contain keyword 3} \} \]

\[ M \] is used to construct the pheromone matrix of the dimensions \( a \times a \), where \( a \) is the cumulated number of services in \( M \).

At the beginning of each iteration an ant has to select a first service. One way of doing this is to choose a random service of the services in \( M \). Thinking one step further leads to the conclusion that some services are more suitable than others. The probabilities for services that both provide the same keywords but have different costs should not be equal. It appears to be useful to determine a function that evaluates the service's attributes.

\[ f(x, y) = \frac{1}{2} \left( \frac{250}{x} + \frac{y}{v} \right) \]

where

\[ x = \text{costs of a service} \]
\[ y = \text{number of covered keywords} \]
\[ v := \begin{cases} 5, & t > 5 \\ t, & \text{else} \end{cases} \]

\[ T = \text{number of input keywords} \]

Since the costs of a service are limited to at least 250 and no more than 2500 and given that it can only hold 2 to 5 keywords, the function returns 1 for a best suitable service (costs: 250, input completely covered) and 1/20 for the worst (costs: 2500, no keyword covered).

B. Ant Colony Optimization Heuristic

Setup as previously described. For \( n \) iterations:

1) For \( m \) ants:
   - Selection of the first service of the composition using roulette-wheel-selection. The probability of a service is represented by \( f(x, y) \), as shown in the previous section.
   - While the composition does not completely cover the input keywords:
     a) Create a set of all services that contain at least one of the missing keywords.
     b) Calculate the heuristic values for all services in this set. The calculation of these values is explained in 1).
     c) Select a service of the set with roulette-wheel-selection. The probability of a service consists of its heuristic value and the pheromone value on the trail.
   - Spread pheromone on all selected services in dependency of the quality of the found composition. The calculation is shown in 2).

2) Evaporate pheromone

The visibility of the attributes of possible next services is an important part of the algorithm. The heuristic expresses how good a service would match with the current composition. A service is ideal, if it covers as many missing keywords as possible, its costs are minimal and the
composition does not include any of the keywords of it. These three criteria are used in a function shown below:

\[ g(x, y, z) = \frac{1}{3} \left( \frac{x}{y} + \frac{250}{y} + \frac{1}{1+z} \right) \]  

(7)

where
\[ x = \text{number of missing keywords} \]
\[ y = \text{costs of a service} \]
\[ z = \text{number of resulting duplicate keywords} \]

\[ y' := \begin{cases} 5, & t > 5 \\ t, & \text{else} \end{cases} \]

\[ t = \text{number of input keywords} \]

To achieve comparability between compositions it is necessary to define criteria the compositions should fulfill. The result of this intelligent heuristic is an optimum composition that covers all entered keywords and ensures that its cumulated costs and the number of duplicate keywords are at a minimum. Since the first criterion is already met by the algorithm's architecture only costs and duplicate keywords are taken into account. The following function evaluates the quality of a complete composition.

\[ e(x, y) = \frac{1}{2} \left( \frac{250}{x} + \frac{1}{1+y} \right) \]  

(8)

where
\[ x = \text{costs of a service} \]
\[ y = \text{number of duplicate keywords} \]

The result is spread on the path the ant examined.

V. PRACTICAL IMPLEMENTATION

A. Configuration

The settings are carried out centrally in the register Configuration. The behaviour of the algorithm ACO is significant determined by the parameters alpha and beta.

- \( \alpha > \beta \): there is bigger influence on the choice of path, which is more often explored,
- \( \alpha < \beta \): there is bigger influence on the choice of path, which offers better solution,
- \( \alpha = \beta \): there is balanced dependency between quality of the path and degree of its exploration,
- \( \alpha = 0 \): there is a heuristics based only on the quality of passage between consecutive points (ignorance of the level of pheromone on the path).

A Parameter Recommender was developed to find an optimum setting for \( \alpha \) and \( \beta \) regarding the data instance (see Fig. 1 - Fig. 6).

B. Query

The services provided in the cloud environment support the customers by the realization of core demands and business processes. The textual demands of the customer are aligned with the registered keywords. In this case the keywords “4”, “2” and “8” were identified and transferred to the Query GUI. In addition to the automated analysis of textual requirements a manual input of keywords by the user is possible.

By mouse click on the button Execute the search based on ACO is being started. State-of-the-art technologies provided by Java are used for efficient resource usage and parallel processing.

C. Composition

As a result of this search the optimization delivers a dynamic service composition. For this composition the services “5”, “233” and “138” were selected. The business demands specified by the customer and expressed through keywords are fulfilled in this optimization problem regarding the following criteria: 1) maximum number of keywords are covered, 2) minimum overlapping keywords and 3) minimum costs.

D. Benchmark

Fig. 1. Graphical user interface - configuration.

Fig. 2. Graphical user interface - query.

Fig. 3. Graphical user interface - composition.

Fig. 4. Graphical user interface — Benchmark I.
The Benchmark is the analysis tools to compare the solution calculated by ACO. All possible combinations are evaluated by the brute force approach and stored in memory. The data from the benchmark is evaluated statistically and visualized for the purpose of comparison in a tabular overview.

![Fig. 5. Graphical user interface – Benchmark II.](image)

E. Parameter Recommender

Objective of the Parameter Recommender is the intelligent determination of optimum values of the ACO parameters $\alpha$ and $\beta$ concerning solution quality with steady data instance and iterations.

![Fig. 6. Graphical user interface – parameter recommender.](image)

VI. CONCLUSION

An intelligent solution based on nature-inspired swarm intelligence for keyword-based service matching was developed and successfully applied in a cloud environment at an international airport in Germany. The main additional value to existing approaches is the integrated practical solution of keyword-based service matching based on swarm intelligence, intelligent parameter recommendation and an innovative graphical user interface. Using optimum parameter settings this multi-dimensional optimization was carried out in few seconds by the intelligent heuristics based on ACO.

Future steps are additional practical implementations and further research in parameter settings and heuristic dependencies.

Updates are found on http://www.itg-research.net.

REFERENCES


Carsten Mueller holds a M.Sc. degree from the University of Liechtenstein in Vaduz and a Ph.D. degree from the University of Economics (Department Informatics and Statistics) in Prague. He works over 15 years as an independent consultant and trainer in the areas of IT process optimization, object-oriented modelling and IT governance. He is currently the technical head of processes and IT management at a large airport in Germany and a senior lecturer for software engineering, UML and algorithms at the Baden-Wuerttemberg Cooperative State University.

Niklas Kiehne is interested in swarm intelligence with the focus on ant colony optimization and NP-hard problems. He is currently working for a leading global supplier of technology and service in Germany. At the Baden-Wuerttemberg Cooperative State University, he studies applied informatics and he is a member of a research group.