LASA-HEU: Heuristic Approach for Service Selection in Composite Web Services

N. Sasikaladevi
Associate Professor
Department of Computer Applications
Nehru Institute of Engineering and Technology
Coimbatore, TN, India
E-Mail : sasikalade@yahoo.com

Dr. L. Arockiam
Associate Professor
Department of Computer Science
St.Joseph's College
Trichirapalli, TN, India
E-Mail: larockiam@yahoo.com

***Abstract*—Numerous functionally similar services are evolving day by day. Selecting the service which matches exactly with the requirements of the consumer is a tedious task. The QoS-based Service Selection Problem (SSP) is a process of allocating a QoS based exterior web service component to each task of the workflow that describes a composite web service. Hence, the aggregate QoS of the composite web service is the best. It is a planning problem by its nature. This paper provides the brief overview of the heuristic based Service Selection Algorithm (LASA-HEU) for the MMKP form of reliability enforced SSP. This paper also compares the proposed LASA-HEU with the existing heuristic based SSA and proved that the proposed LASA-HEU performs better than the existing heuristic based SSA based on the reliability.**

Keywords-Service Selection, ServiceOrientation, MMKP, LASA-HEU

I. INTRODUCTION

Service-Orientation (SO) is a design standard which intends realizing distributed software solutions as services [1]. A service is well suited for a distributed system because; a component in the description of a service is not only designed by the functional interface, but also of the activities and the non-functional properties [2]. This permits for modular approach and loose coupling.

Service-Oriented Computing (SOC) has emerged as a modern computing paradigm following the SO potency. SOC consists of numerous concepts and technologies to recognize distributed software solutions as services, ranging from SOA to Cloud Computing (CC) and beyond. The main purpose of SOC is to allow a fast and low-priced development and composition of distributed applications, following the vision of a service-based infrastructure where distributed applications are easily assembled from existing services [3].

Service Selection (SS) is a challenging task in SO. Numerous functionally similar services are evolving day by day. Selecting the service which matches exactly with the requirements of the consumer is a tedious task. The selection process is usually twofold: The first step considers only the services that exactly match the functional criteria. The second step selects the best one by ranking those services according to which extent they satisfy the non-functional criteria.

Functional matching primarily operates on semantics. Once a service consumer specifies the desired functionality semantically, it is matched against semantic descriptions of available services [4]. Exact matches are prevalent in SS, because, if a service does not do what a consumer wants, then it is meaningless to call that service.

Non-functional matching conversely revolves around NFPs which can basically be considered as constraints over the offered functionality [5]. The NFPs of a service are generally defined as a contract between the service provider and service consumer called Service Level Agreement (SLA). The standards for SLAs are defined in Web Service Level Agreements (WSLA) [6] and Web Service Modeling Language (WSML) [7].

A. Service Selection Problem(SSP)

Several abstract services are composed to form an abstract workflow. When each service is replaced with a concrete deployment, the abstract workflow is mapped to a concrete workflow. The service selection optimization procedure transforms the abstract service workflow into a concrete deployment workflow.

Mathematically, this is modeled as a Multi-dimensional Multi-choice Knapsack Problem (MMKP). In MMKP, a knapsack has to be packed with items that are classified into mutually exclusive object groups, each containing several different objects. The knapsack itself has a set of constraints. The objective of MMKP is to select precisely one item from each class in such a way that it maximizes the total profit of the collected objects, subject to resource constraints of the knapsack [8].

In the same way, the Reliability enforced SSP is to choose one deployment for each abstract service to build the concrete workflow. The workflow should have defined QoS constraints (Price, Response time and Reputation) and maximize the reliability.

II. RELATED WORK

Diana Comes et al. [9] proposed multi objective based heuristic algorithm for the selection of services in service orchestrations. The OPTIM_HWeight algorithm is based on weight factors and the OPTIM_PRO algorithm is based on priority factors for performing the service selection. The OPTIM_HWeight make use of a specific heuristic function called HWeight.

Diana Comes et al. [9] proposed another multi objective based heuristic service selection algorithm OPTIM_PRO. The OPTIM_PRO heuristic algorithm estimates priority factors and applies the objective function to sort the variants.

Rong Wang et al. [10] proposed the fast heuristic algorithm for SSP. In this work, the utility function is described to reflect the QoS parameters of the service selection model. And then, the service selection model is mapped into MMKP form. It is the Simple Heuristic that is used to solve the MMKP

Tao Yu et al. [10] proposed WS-HEU algorithm for web service selection for composite web service. WS-HEU intended to find solutions for MMKP. WS-HEU is extended from the algorithm HEU [11][12] that uses an initial feasible solution by always searching for the lowest utility item in each service class. The search is time-consuming. WS-HEU prunes out more infeasible items from each class and finds a feasible solution in a shorter time.

A. Performance Comparison of Multi Objective Heuristic based Service Selection Algorithms

The well known multi objective based heuristic approaches are explained in the previous sub section. All these algorithms are analyzed based on the approach incorporated, QoS factors incorporated and the optimality of the result achieved. The Table I shows the results.

TABLE I
TYPE SIZES FOR PAPERS COMPARISON OF MULTI OBJECTIVE HEURISTIC BASED SERVICE SELECTION ALGORITHMS

S. No.	**Heuristic Models**	**Approach Used**	**QoS Factors**	**Optim-ality**
1	OPTIM_HWeight [9]	Gradient descent Based	Reliability Response Time Cost Availability	95%
2	OPTIM_PRO [9]	Priority Based	Reliability Response Time Cost Availability	95%
3	F-HEU[10]	Penalty Based	Response Time Cost	93%
4	WS_HEU[11]	Khan's Approach	Response Time Cost Availability	97%

Diana's Models OPTIM_HWeight and OPTIM_PRO incorporate reliability, response time, cost and availability as the QoS factors. OPTIM_HWeight uses the gradient descent approach. The result of the $f_{HWeight}$ function is used to deliver a score for the candidate service selection versus the current service selection; a higher value is ranked higher. This approach is referred as a gradient ascent. OPTIM_PRO is based on Priority approach. Candidates are ranked based on the utility value. Higher priority is given to service candidates with highest utility value. Highly ranked candidates are having higher probability of selection. F-HEU incorporated response time and cost as the QoS factors. F-HEU is penalty based approach. Penalty is assigned to infeasible candidates. Probability of selecting the infeasible candidates is low. WS-HEU incorporates response time, cost and availability as the QoS factors. WS-HEU is based on Khan's Approach. Khan's approach is based on the concept of one upgrade and one or more downgrades.

In order to improve the optimality further, the novel multi objective heuristic based service selection algorithm named LASA-HEU is proposed in this paper. It is an improved version of WS-HEU

III. PROPOSED MODEL

The proposed Heuristic Service Selection Algorithm LASA-HEU is based on [11] [12]. The vital change in the proposed algorithm is the elimination of downgrades in order to reduce the time taken for execution and to avoid down fall on the reliability. The LASA-HEU is implemented with iterative upgrades. The proposed algorithm is explained in the subsequent sections.

A. LASA-HEU Overview

The LASA-HEU for reliability enforced SSP for composite web service is based on the following concepts: it commences with finding a feasible solution. At first, it finds a solution from each service group; the service candidate with the smallest reliability is selected. If it is not feasible, it tries with an iterative procedure for replacing one chosen item with another to find a feasible solution. It uses Toyoda's concept of aggregate resource [13] for selecting service candidate from each service group.

Here, the predominant task is to use weighted penalization for the not-yet-selected service candidates depending on the current value of QoS constraints (Response time-T, Cost-C and Vendor Reputation-V) and the QoS requirement of service candidates. It includes a high penalty value for a heavily used QoS factor, and a small penalty factor for a lightly used QoS factor. It uses iterative improvement of the solution by using exchange of selected items. In each exchange, the status of two candidates (one selected and the other not-selected) in a group are swapped.

The penalty approach used in this LASA-HEU is to find the optimal or near-optimal solution by upgrading the feasible solution. The algorithm works if it starts upgrading from any feasible solution. The popular heuristic approach proposed by Mos et al. [14] finds a feasible solution starting from the highest-valued service candidates, whereas in LASA-HEU lowest-valued service candidates are selected for initial feasible solution.

Experimental results show that Moser approach does not perform well for SSPs. It is easy to find a feasible solution by checking the lowest valued items in the cases where the QoS follows the monotonous feasibility property. So, LASA-HEU starts from the lowest valued items and tries to find a feasible solution by upgrading if the initial pick is a valid approach. Following are the 2 steps involved in LASA-HEU.

i. Finding a feasible solution:

Step 1 starts with a pick which includes the lowest-valued service candidate from each service group. If this choice is feasible then proceed to the next step for iterative improvement. If the initial pick is not feasible, find the QoS with the highest feasibility factor. The next step is to find a higher valued service candidate such that if this service candidate is picked then:

a) It reduces the feasibility factor of QoS,
b) It does not increase the feasibility factor of any previously infeasible QoS, and

c) It makes any previously feasible QoS as infeasible.

If such a service candidate cannot be found, the procedure returns with “no solution found”. If more than one candidate is found, then pick the candidate that maximizes the savings in aggregate QoS saving. If the current solution is feasible, proceed to step 2. If the current solution is not feasible, repeat this step until either a feasible solution is reached or the heuristic is returned with “no solution found”.

ii. Iterative improvement using feasible upgrades:

Try to improve the solution using iterative search for feasible upgrades. In each iteration, finding a feasible upgrade involves the following steps:

a) Estimate the aggregate QoS saving for all feasible upgrades.
b) If there exists at least one feasible upgrade which provides savings in aggregate QoS, then LASA-HEU chooses the upgrade which maximizes the savings in aggregate QoS. It is reasonable, because it improves the feasibility of the resources and frees up QoS thus making room for possibly more feasible upgrades.

If there is no feasible upgrade which provides savings in aggregate QoS, then LASA-HEU chooses the upgrade which maximizes the value gain per unit of extra aggregate QoS.

LASA-HEU starts with initial solution. It calculates the aggregate resource saving using Toyoda’s principle [13]. Based on this value, the flow proceeds with either upgrade or exchange operation. These steps are repeated until the convergence condition is reached.

IV. ANALYSIS ON LASA-HEU

LASA-HEU is investigated by conducting experiments with a composite service samples. 10 different composite service samples are considered for this experimentation. Each composite service sample includes tasks connected by basic work flow patterns. Each task has a candidate service set, and each candidate service has estimated reliability value (using AROSA[1] explained [18]) and three QoS constraints which are service price (C), response time (T), and level of reputation (U).

The optimization objective is to maximize the reliability with respect to the QoS constraints cost, execution time and vendor reputation. The execution platform of the proposed LASA-HEU algorithm is a HP Pavilion G6 System with Microsoft Windows 7 Home Premium Operating System, Intel Core i3-380M Processor with the speed of 2.53 GHz, Dual Core System, L3 cache-3 MB, 64 bit computing, 8 GB RAM, DDR3 SDRAM and with the Hard disk with the capacity of 500 GB HDD/5400 rpm. The algorithm is implemented using Visual C++ and executed in the Microsoft Visual C++ Express Edition.

[1] AROSA is the reliability evaluation framework for the web services[18]. It is used here to evaluate the reliability of each service candidate in the service groups.

A Population Involved on LASA-HEU

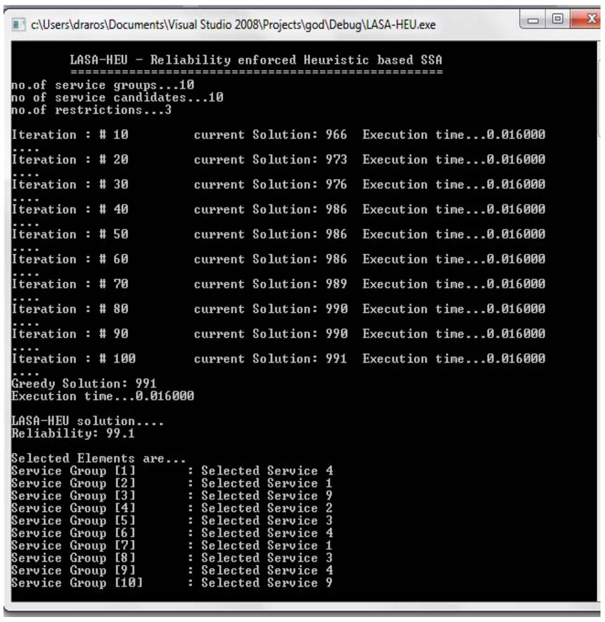
Several experiments are conducted to evaluate the efficiency of the proposed algorithm. Different populations are used to evaluate the LASA-HEU. Service Groups ranges from 10 to 50. For each service group, service candidate’s ranges from 10 to 50. In the first experiment, the runtime of the LASA-HEU with different population sizes (respectively 10, 20, 30, 40 and 50) is compared while the size of candidate varies from 10 to 50.

TABLE II
SAMPLE DATA SET FOR LASA-HEU

$N_g = 10$ $N_c = 10$										
Max. Cost = 50 $ Max. Res. Time = 50 time unit Reputation Level Expected = 50 Level										
	SG$_1$	SG$_2$	SG$_3$	SG$_4$	SG$_5$	SG$_6$	SG$_7$	SG$_8$	SG$_9$	SG$_{10}$
SC$_1$	97	**99**	82	93	78	91	**99**	94	87	91
	9	**8**	4	6	0	7	**6**	3	6	3
	7	**7**	2	7	7	8	**6**	0	6	9
	1	**6**	2	4	3	2	**4**	1	1	2
SC$_2$	90	93	91	**99**	74	94	93	94	98	92
	6	8	9	**2**	8	3	1	4	3	7
	3	5	1	**4**	6	4	5	7	1	1
	3	5	4	**3**	3	5	4	3	2	3
SC$_3$	85	85	74	96	**99**	98	92	**99**	90	98
	2	8	7	8	**3**	9	1	**9**	9	6
	5	1	5	0	**3**	8	2	**6**	3	0
	2	4	3	5	**5**	3	1	**5**	1	0
SC$_4$	**99**	96	94	94	78	**99**	97	92	**99**	92
	7	1	6	7	5	**3**	0	5	**3**	9
	1	7	3	9	4	**9**	6	6	**6**	1
	5	3	1	2	3	**2**	3	4	**8**	2
SC$_5$	91	76	75	75	90	99	98	94	91	93
	9	6	5	5	7	7	8	8	8	5
	9	8	7	8	3	1	6	5	7	3
	1	4	3	2	3	6	3	2	3	1
SC$_6$	72	95	90	82	63	82	90	77	82	97
	1	6	0	1	7	8	7	0	5	3
	3	1	5	4	9	0	1	1	1	1
	2	3	2	4	3	2	1	0	8	1
SC$_7$	96	98	92	94	73	94	92	87	80	97
	2	7	8	6	3	5	8	1	4	9
	7	6	2	7	8	8	5	6	9	0
	1	2	0	9	8	4	2	1	4	3
SC$_8$	75	76	99	84	96	76	84	95	91	94
	0	5	6	1	4	7	0	6	7	7
	4	6	9	0	6	6	3	7	0	6
	1	3	1	3	5	6	3	4	3	3

SC_9	92	90	**99**	92	80	92	93	96	88	**99**
	5	4	**3**	1	4	4	6	6	4	**1**
	2	5	**0**	5	6	0	4	1	2	**8**
	0	1	**5**	0	1	2	5	4	8	**7**
SC_{10}	97	93	95	84	99	93	81	97	89	90
	0	7	0	8	9	3	0	0	4	6
	0	4	9	5	6	5	1	3	2	1
	6	7	4	9	0	7	6	6	8	6

Sample data set for the LASA-HEU is shown in Table II. Cost, response time and the reputation of each service candidate is estimated by using the method proposed in [18]. Fig. 1 shows the output for the data set given in Table II. The output screen shows the reliability in every iteration.



```
c:\Users\draros\Documents\Visual Studio 2008\Projects\god\Debug\LASA-HEU.exe

        LASA-HEU - Reliability enforced Heuristic based SSA
        ================================================
no.of service groups...10
no of service candidates...10
no.of restrictions...3

Iteration : # 10        current Solution: 966  Execution time...0.016000
....
Iteration : # 20        current Solution: 973  Execution time...0.016000
....
Iteration : # 30        current Solution: 976  Execution time...0.016000
....
Iteration : # 40        current Solution: 986  Execution time...0.016000
....
Iteration : # 50        current Solution: 986  Execution time...0.016000
....
Iteration : # 60        current Solution: 986  Execution time...0.016000
....
Iteration : # 70        current Solution: 989  Execution time...0.016000
....
Iteration : # 80        current Solution: 990  Execution time...0.016000
....
Iteration : # 90        current Solution: 990  Execution time...0.016000
....
Iteration : # 100       current Solution: 991  Execution time...0.016000
....
Greedy Solution: 991
Execution time...0.016000

LASA-HEU solution....
Reliability: 99.1

Selected Elements are...
Service Group [1]       : Selected Service 4
Service Group [2]       : Selected Service 1
Service Group [3]       : Selected Service 9
Service Group [4]       : Selected Service 2
Service Group [5]       : Selected Service 3
Service Group [6]       : Selected Service 4
Service Group [7]       : Selected Service 1
Service Group [8]       : Selected Service 3
Service Group [9]       : Selected Service 4
Service Group [10]      : Selected Service 9
```

Fig. 1 Sample output of LASA-HEU

LASA-HEU is compared with existing well known heuristic based service selection algorithm WS-HEU proposed by Tao Yu [Tao, 07]. Execution time of LASA-HEU is 2 times lesser as compared to the execution time of WS-HEU. The optimality rate of LASA-HEU is approximately 2% better than the WS-HEU.

V. Conclusion

Reliability enforced SSP is solved by using the proposed algorithm LASA-HEU. Toyoda's aggregate resource saving concept is implemented in LASA-HEU. The special selection operation is designed to select the feasible candidates. The LASA-HEU is compared with WS-HEU which is a well known heuristic based service selection algorithm. The execution time of the LASA-HEU is 2 times lesser than the WS-HEU. The average reliability of WS-HEU is in the range 98 to 99. The average reliability of LASA-HEU is in the range 99 to 100.

References

[1] Erl T (2007). *SOA Principles of Service Design,* Prentice Hall PTR.

[2] MacKenzie C. M, Laskey K., McCabe F., Brown P. F., Metz R (2006). Reference Model for Service Oriented Architecture 1.0, OASIS Standard. http://docs.oasis-open.org/soa-rm/v1.0/, 2006.

[3] Papazoglou M.P, Traverso P, Dustdar S, Leymann F (2003). Service-oriented computing, Communications of the ACM, 46, 25–28.

[4] Paolucci M, Kawamura T, Paynen T.R, Sycara K.P (2002). Semantic matching of web services capabilities. Lecture Notes in Computer Science, 23, 333–347.

[5] Osullivan J, Edmond D, Ter Hofstede A (2002). What's in a service? Towards accurate description of non-functional service properties, Distributed and Parallel Databases, 12(23), 117–133.

[6] Ludwig H, Keller A, Dan A, King R, Franck R (2003). Web service level agreement (WSLA) language specification: Version 1.0. http://www.research.ibm.com/wsla/WSLASpecV1

[7] De Bruijn, Lausen, Reto Krummenacher, Axel Polleres, Livia Predoiu, Michael Kifer, Dieter Fensel, Ioan Toma, Nathalie Steinmetz, and Mick Kerrigan (2007). The web service modeling language WSML, http://www.wsmo.org/TR/d16/d16.1/v0.3/, 2007.

[8] Gen Cheng (2000). Genetic Algorithms and Engineering Optimizations, Wiley.

[9] Diana C, Harun B , Roland R, Michael Z, Kurt G. (2010). Heuristic Approaches for QoS based Service Selection. Springer LNCS, 441-455.

[10] Tao Yu, Lin K.J (2005). A Broker-Based Framework for QoS-aware Web service composition. Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service, 22–29.

[11] Khan S, Li K.F, Manning E G, Akbar M.M (2002), Solving the knapsack problem for adaptive multimedia systems, Dissertation, Informatics University, 2(1), 157–178.

[12] Khan S (1998), Quality adaptation in a multisession multimedia system: Model, algorithms and architecture, Ph.D. thesis, University of Victoria.

[13] Toyoda Y (1975). A Simplified Algorithm for Obtaining Approximate Solution to Zero-one Programming Problems, Management Science, 21, 1417–1427.

[14] Moser M, Jokanovic D. P, Shiratori N (1997), An algorithm for the multidimensional multiple-choice knapsack problem, IEEE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 80 (3), 582–589.

[15] N. Sasikaladevi, L. Arockiam, "Genetic Approach for Service Selection Problem in Composite Web Service", *International Journal of Computer Applications (IJCA)*, ISSN: 0975-8887, Vol. 44, No.14, pp.22-30, April 2012.

[16] L. Arockiam, N. Sasikaladevi, "Simulated Annealing based Service Selection Algorithm for Composite Web Service", *International Journal of Advanced Research in Computer Science (IJARCS*), ISSN: 0976-5697, Vol.3, No.2, pp.132-141, March 2012.

[17] L. Arockiam, N. Sasikaladevi, "Simulated Annealing versus Genetic Based Service Selection Algorithms", *International Journal of u- and e-Service, Science and Technology* (IJUESST), ISSN: 2005-4246, Science and Engineering Support Society(SERC), Korea, Vol.5, No.1, pp.35-50, February 2012.

[18] L. Arockiam, N. Sasikaladevi, "Estimating the Reliability of Composite Web services- Service Consumer Perspective", *International Journal of Advanced Research in Computer Science (IJARCS)*, ISSN: 0976-5697, Vol.3, No.1, pp.267-274, January 2012.