A Paper on COMPOSITE SERVICE SELECTION

J. Gowtham Kumar [#1], B. Sudheer Kumar [*2], K.Narasimhulu [#3]

[1#]Student of MCA, RGM College,
Nandyala, Kurnool(Dt), Andhra Pradesh.
[3]Assistant Professor ,Dept of CSE*
RGM College of Eng&Tech, Nandyal, Andhra Pradesh
[1#]gowtham77599@gmail.com
[3#] narasimhulu.kolla@gmail.com
*[2**]Assistant Professor, Dept of MBA*
Vaagdevi Institute of Technology & Science, Proddatur, Andhra Pradesh
[2]bskmba06@gmail.com

Abstract— **Service composition is very important now days because integrating all the related services and providing good service it plays a role. Giving services is not important to provide best match for the service is the important one and also providing the functionalities like time, fee etc. We propose to exploit the dominance relationship among service providers to find a set of "best" possible composite services, referred to as a composite service skyline. Here we proposed composite service skyline to reduce the searching space instead of composing all services for this we proposed efficient algorithms. We conduct a comprehensive analytical and experimental study to evaluate the effectiveness, efficiency, and scalability the composite skyline computation approaches.**

Keywords— **Skyline Services, QoS Broker Architecture, Algorithms for the Combinatorial Model, WS_ HEU Algorithm.**

I. INTRODUCTION

In composition of skyline important objective is to enable the interoperability among different web applications and software in different platforms. Daily number of services is increase so finding the best one is very critical. These services gives a challenge to the developers for service composition means that selection of appropriate service that satisfies the customer need as quality of web service.

Now a day's business process out sourcing is improved so they mainly focus on their core activities. In that web users wants composition of services to achieve more complex tasks which can't achieved by a single web service. Here we use Service-oriented Architecture paradigm, composite applications are specified as abstract processes composed of a set of abstract services. Then at run time a best service selected and used.

Here we use a web application for finding the best used car offers. The user enters the details of the car which he/she want. Then the system returns the set of best offers with credit and insurance offer for each car in the list. Here the composed application works as a web service to the user. Here the API programmatically integrated into their (user) web applications using a Mash up tool.

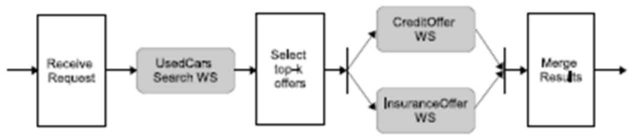

Fig: 1 Service Composition

Here some tasks like Used Cars and Credit Offers are shown in Gray color and these services are outsourced and integrated via web services. For these outsourced tasks, multiple services may be available providing the required functionality but with different QoS values. Users are typically unaware of the involved services, and they specify their QoS requirements in the SLA in terms of end-to-end QoS constraints (e.g. average end-to-end response time, minimum overall throughput, maximum total cost). The goal of QoS-based service composition is to select one service or service configuration for each outsourced task such that the aggregated QoS values satisfy all the application level QoS constraints.

II.SKYLINE SERVICES

Here the goal is to select a set of services, one from each service class, that maximize the overall utility, while satisfying all the specified constraints. However, not all services are potential candidates for the solution. The main idea in our approach is to perform a skyline query on the services in each class to distinguish between those services that are potential candidates for the composition, and those that cannot possibly be part of the final solution. The latter can effectively be pruned to reduce the search space.

Given a set of points in a d-dimensional space, a skyline query selects those points that are not dominated by any other point. A point Pi is said to dominate another point Pj, if Pi is better than or equal to Pj in all dimensions and strictly better in at least one dimension. Intuitively, a skyline query selects the “best” or most “interesting” points with respect to all dimensions. In this work, we define and exploit dominance relationships between services based on their QoS attributes. This is used to identify services in a service class that are dominated by other services in the same class.

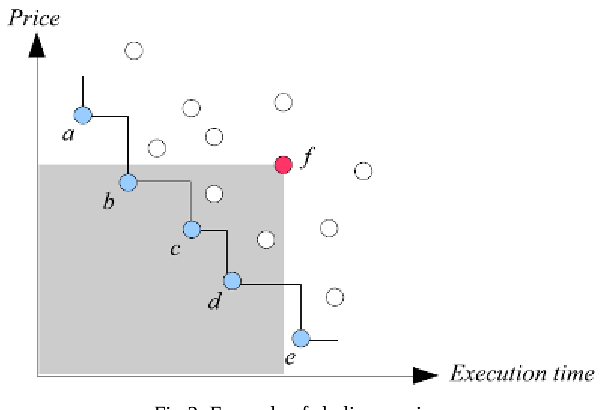

Fig 2: Example of skyline services

Figure 2 shows an example of skyline services of a certain service class. Each service is described by two QoS parameters, namely execution time and price. Hence, the services are represented as points in the 2-dimensional space, with the coordinates of each point corresponding to the values of the service in these two parameters. We can observe that the service a belongs to the skyline, because it is not dominated by any other service, i.e. there is no other service that offers both shorter execution time and lower price than a. The same holds for the services b, c, d and e, which are also on the skyline. On the other hand, service f is not contained in the skyline, because it is dominated by the services b, c and d.

Service-Oriented Architecture (SOA) provides a flexible framework for service composition. Using standard-based protocols (such as SOAP and WSDL), composite services can be constructed by integrating atomic services developed independently. Algorithms are needed to select service components with various QoS levels according to some application-dependent performance requirements. We design a broker-based architecture to facilitate the selection of QoS-based services. The objective of service selection is to maximize an application-specific utility function under the end to- end QoS constraints. The problem is modeled in two ways: the combinatorial model and the graph model. The combinatorial model defines the problem as a multidimensional multi-choice 0-1 knapsack problem (MMKP). The graph model defines the problem as a multi-constraint optimal path (MCOP) problem.

III. QoS BROKER ARCHITECTURE

We have designed a QoS service broker called *Q*Broker, which acts as an external, independent broker entity that can help users construct composite services. The goal of Qbroker is to help users select the best services for their composite service process before invocation. A detailed presentation on the QBroker architecture can be found in Figure 3.

- Process plan. An abstract process that defines a flow of component functions and their relationships. A process plan is designed to fulfill a user’s request.
- Function graph. This is a functional graph which includes all process plans that can fulfill a user’s

request. Each node of the graph is a component function. A start node and an end node are added to the function graph.

- Service candidate graph. A service candidate graph is built from a functional graph and it includes a path for all combinations of function nodes that can be used to fulfill the service request. The executable composite service is created by services selection based on a service candidate graph.

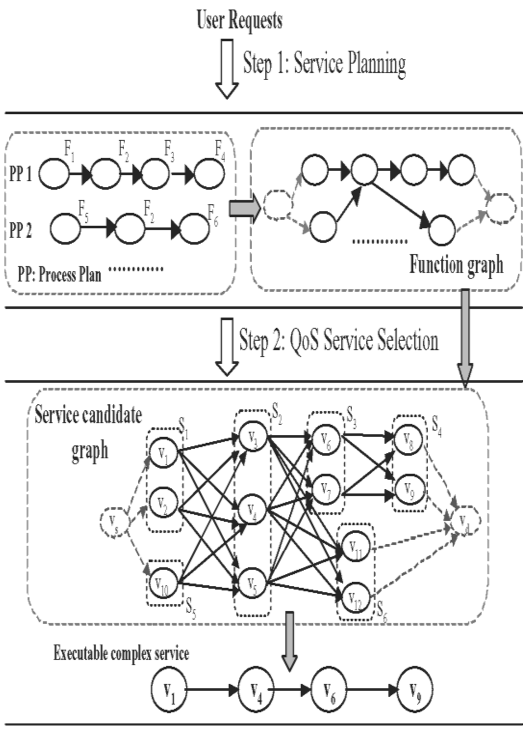

Fig 3: QoS Web service composition

IV.ALGORITHMS FOR THE COMBINATIONAL MODEL

For a composite service that has N service classes (S1, S2. . . SN) in a process flow plan and with m QoS constraints, we map the service selection problem to a 0-1 multidimensional multichoice knapsack problem (MMKP). MMKP is defined as follows. Suppose there are N object groups, each has li ($1 \leq i \leq N$) objects. Each object has a profit pij and requires resource rij = ($r1^{ij}$. . . rm_{ij}). The amount of resources available in the knapsack is R = (R1. . . Rm). MMKP is to select exactly one object from each object group to be placed in the knapsack so that the total profit is maximized while the total resources used are less than the resources available.

The QoS service selection problem is to select one service candidate from each service class to construct a composite service that meets a user's QoS constraints and maximizes the total utility. The QoS service selection problem is mapped to MMKP as follows.

- Each service class is mapped to an object group in MMKP.
- Each atomic service in a service class is mapped to an object in a group in MMKP
- The QoS attributes of each candidate are mapped to the resources required by the object in MMKP.
- The utility a candidate produces is mapped to the profit of the object.
- A user's QoS constraints are considered as the resources available in the knapsack.

V.WS_ HEU ALGORITHM

The computation time for BBLP grows exponentially with the size of the problem. This may not be acceptable for Qbrokers that need to make runtime decisions. Heuristic algorithms may be useful to find feasible solutions in polynomial time. We use a heuristic algorithm WS HEU to find solutions for MMKP. Figure4 shows the flow of the WS_ HEU algorithm and the individual functions (F1 to F6) used in Figure4. The algorithm has three main steps.

(1) Find an initial feasible solution. For each service class Si, WS_ HEU selects a service ρ_i that has $\min_j$ {maxα{ qα ij/Qαc }} in the class. It then checks the feasibility of the initial solution. If the solution is infeasible, the algorithm iteratively improves the solution by replacing the service ρ'_i with the largest saving of aggregated QoS among all service classes. The service replacement continues until a feasible solution is found (or else the algorithm fails).

(2) Improve the solution by feasible upgrades. Among all classes, WS _HEU finds service ρ'_i to replace $\rho' i$ for Si to get a higher utility without violating the constraint requirements. The service replacement criterion is based on. If no such

service can be found, WS_HEU picks the one that maximizes *pij*.

(3) Improve the solution by infeasible upgrades followed by downgrades. Performing only feasible upgrades may reach a local optimal in the search space. To achieve the global optimal, WS_HEU further improves the solution by using F5 to select theΔservice that maximizes p'_{ij}. This replacement makes the solution infeasible.

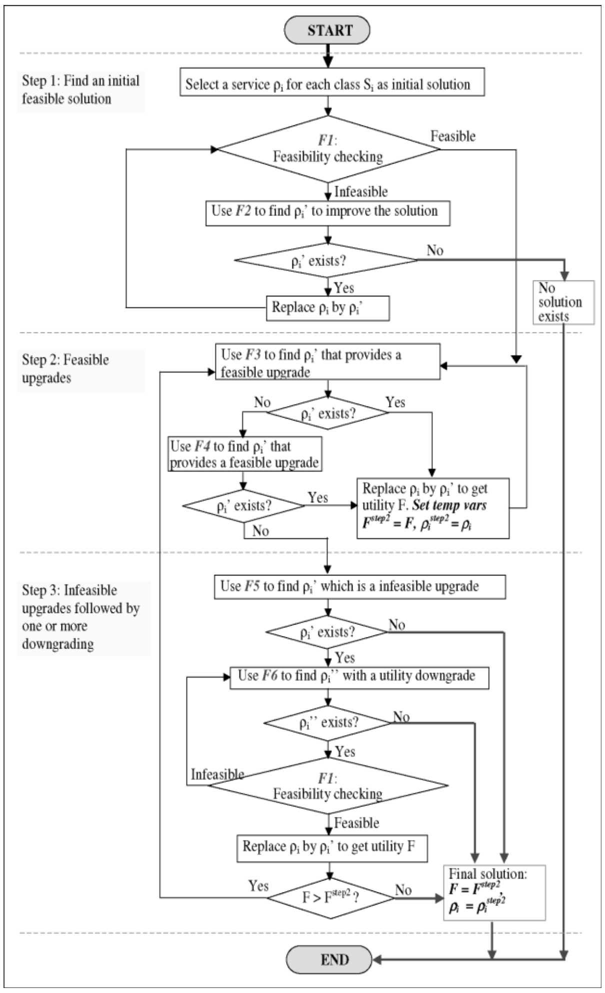
START
Step 1: Find an initial feasible solution
Select a service ρi for each class Si as initial solution
F1: Feasibility checking
Feasible
Infeasible
Use F2 to find ρi' to improve the solution
ρi' exists?
No
Yes
Replace ρi by ρi'
No solution exists
Step 2: Feasible upgrades
Use F3 to find ρi' that provides a feasible upgrade
No
ρi' exists?
Yes
Use F4 to find ρi' that provides a feasible upgrade
ρi' exists?
Yes
No
Replace ρi by ρi' to get utility F. Set temp vars F^step2 = F, ρi^step2 = ρi
Step 3: Infeasible upgrades followed by one or more downgrading
Use F5 to find ρi' which is a infeasible upgrade
ρi' exists?
No
Yes
Use F6 to find ρi'' with a utility downgrade
ρi'' exists?
No
Yes
Infeasible
F1: Feasibility checking
Feasible
Replace ρi by ρi' to get utility F
Yes
F > F^step2 ?
No
Final solution: F = F^step2, ρi = ρi^step2
END

Fig 4: Algorithm structure for WS_ HEU

VI.WEB INFORMATION SYSTEM ARCHITECTURE AND RELATED WORK

Modern Web information systems feature architecture like the one roughly sketched in figure 1. Using a (mobile) client device the user poses a query. Running on an application/Web server this query may be enriched with information about a user (e.g. taken from stored profiles) and will be posed to a set of Internet sources. Depending on the nature of the query different sources can be involved in different parts of the query, e.g. individual sources for traffic jams or weather information. Collecting the individual results the combining engine runs an algorithm to compute the overall best matching objects. These final results have then to be aggregated according to each individual user's specifications and preferences. After a transformation to the appropriate client format (e.g. using XSLT with suitable stylesheets) the best answers will be returned to the user.

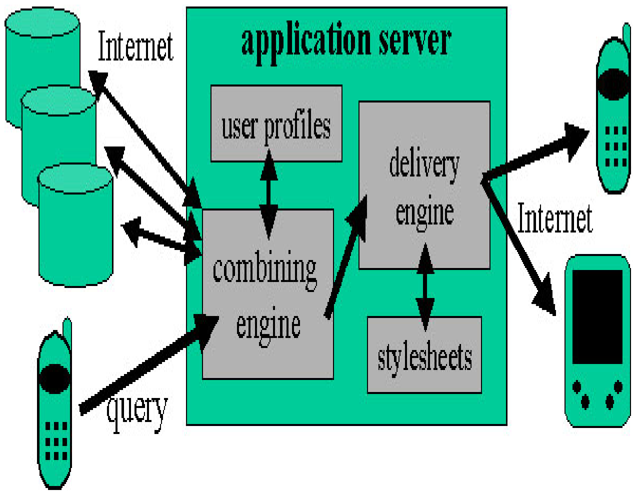

Fig 5: Web information architecture

The first area to address such a distributed retrieval problem was the area of 'top k retrieval' over middleware environments; especially for content based retrieval of multimedia data these techniques have proven to be particularly helpful. Basically all algorithms distinguish between different query parts (sub queries) evaluating different characteristics, which often have to be retrieved from various subsystems or web sources. Each subsystem assesses a numerical score value (usually normalized to [0, 1]) to each object in the collection. The middleware algorithms use two basic kinds of accesses that can be posed: there is the iteration over the best results from one source (or with respect to a single aspect of the query) called a 'sorted access' and there is the so-called 'random access' that retrieves the score value with respect to one source or aspect for a certain given object.

The physical implementation of these accesses always strongly depends on the application area and will usually differ from system to system. The gain of speeding up a single access (e.g. using a suitable index) will of course complement the total runtime improvement by reducing the overall number of accesses. Therefore minimizing the number of necessary object accesses and thus also the overall query runtimes is tantamount to build practical systems (with real-time constraints). However, all these top k retrieval systems relied on a single combining function (often called 'utility function') that is used to compensate scores between different parts of the query. Being worse in one aspect can be compensated by the object doing better in another part. However, the semantic meaning of these (user provided) combining functions is unclear and users often have to guess the 'right' weightings for their query. The area of operations research and research in the field of human preferences has already since long criticized this lack in expressiveness.

A more expressive model of non-discriminating combination has been introduced into the database community. The 'skyline' or 'Pareto set' is a set of no dominated answers in the result for a query under the notion of Pareto optimality. The typical notion of Pareto optimality is that without knowing the actual database content, there can also be no precise a-prior knowledge about the most sensible optimization in each individual case (and thus something that would allow a user to choose weightings for a compensation function). The Pareto set or skyline hence contains all best matching object for all possible strictly monotonic optimization functions. An example for skyline objects with respect to two query parts and their scorings S1 and S2 is shown in figure 2. Each database object is seen as a point in multidimensional space characterized by its score values. For instance objects ox = (0.9, 0.5) and oy = (0.4, 0.9) both dominate all objects within a rectangular area (shaded). But ox and oy are not comparable, since ox dominates oy in S1 and oy dominates ox in S2. Thus both are part of the skyline.

VII.CONCLUSION

Here we tell about the best and efficient service composition and some algorithms and methods to achieve this composition. In near feature it is help full to develop more and more composition of services.

References

- W.-T. Balke, U. Guntzer, and J.X. Zheng, "Efficient Distributed Skylining for Web Information Systems," Proc. Int'l Conf. Extend- ing Database Technology (EDBT), 2004.
- T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-End Qos Constraints," ACM Trans. Web, vol. 1, no. 1, article 6, 2007.
- M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for Qos-Based Web Service Composition," Proc. 19th Int'l Conf. World Wide Web (WWW)), 2010.
- D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. ACM Trans. on Database Systems, 30(1):41–82, 2005.
- R. Parra-Hernandez and N. J. Dimopoulos. A new heuristic for solving the multi choice multidimensional knapsack problem. IEEE Trans. on Systems, Man, and Cybernetics, Part A, 35(5):708–717, 2005
- KHAN, S. 1998. Quality adaptation in a multisession multimedia system: Model, algorithms and architecture. Ph.D. dissertation, Department of ECE, University of Victoria.
- KHAN,S.,LI,K.F.,MANNING,E.G.,AND AKBAR,M. 2002. Solving the knapsack problem for adaptive multimedia systems. Studia Informatica Universalis 2, 1, 157