# Blackboard Meets Dijkstra for Resource Allocation Optimization

Christian Vorhemus and Erich Schikuta

Faculty of Computer Science, University of Vienna
Währingerstr. 29, A-1090 Vienna, Austria
{christian.vorhemus,erich.schikuta}@univie.ac.at

**Abstract.** This paper presents the integration of Dijkstra's algorithm into a Blackboard framework to optimize the selection of web resources from service providers. The architectural framework of the implementation of the proposed Blackboard approach and its components in a real life scenario is laid out. For justification of approach, and to show practical feasibility, a sample implementation architecture is presented.

**Keywords:** Web-service selection · Resource allocation optimization · Blackboard method · Dijkstra algorithm

## 1 Introduction

With the advent of cloud computing, where resources are provisioned on demand as services on a pay-per-use basis, the proper selection of services to execute a given workflow is a crucial task. For each service or functionality, a range of concrete services may exist, having identical functional properties, but differing in their non-functional properties, such as cost, performance and availability.

In literature, the challenge of selecting service deployments respecting non-functional properties is widely known as the QoS-aware service selection problem [9]. Given an input workflow with specified abstract services (functionality), select a concrete deployment (out of several possible ones) for each abstract service in such a way that a given utility function is maximized and specified constraints are satisfied. Mathematically, this can be mapped to a multi-dimension, multi-choice knapsack problem that is known to be NP-hard in the strong sense [9]. In reality, we have also to cope with dynamic changes of services and their characteristics during the workflow execution.

In [8] we proposed a blackboard approach to automatically construct and optimize workflows. We pointed out the problem of changing conditions during the execution of the algorithm. In a distributed environment like the Web, the availability of all necessary services to complete a workflow is not guaranteed. Furthermore it is also possible that users change their configuration. To consider changed condition, we used the A-* algorithm which needs not to know all services at the beginning of the execution and allows for dynamic adaptation. A fundamental description of how services can be qualified is presented in [5].

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

Among an UML-based approach to classify QoS-Attributes, also a detailed description of how services can be combined is given.

This paper presents a novel approach for the optimization of web service selection.Hereby, we propose an artificial intelligence approach by mapping the service selection problem to a graph representation and to apply a combination of Dijkstra's algorithm and Blackboard method for optimization.

The layout of the paper is as follows: The Blackboard method and its optimization approach are presented in section 2. In section 3 we depict the architecture for the implementation of the proposed optimization framework for real world scenarios, where all our theoretical findings are knot together. The paper closes with a conclusion of the findings and look-out to further research.

## 2 The Blackboard Approach

The blackboard method [2] originally comes from artificial intelligence and uses different expert knowledge resources taking part in a stepwise process to construct solutions to given problems. The Blackboard framework consists of four different elements:

- A *global blackboard* representing shared information space considering input data and partial solutions, where different experts collect their knowledge and form the partial solutions to a global optimal solution.
- A *resource* is any kind of service which provides functionality that is needed to finish a subtask of a workflow.
- *Agents* are autonomous pieces of software. Their purpose is finding suitable resources for the Blackboard. An Agent can be a service too and therefore also be provided from external sources.
- *Controller:* There are different kinds of controlling components (see 3). Their main purpose is controlling the start of the algorithm, managing the brokers and bringing the results back to the user.

The Blackboard approach expands (combines) promising resource offerings (combinations) step by step, which are stored in the OpenList. All already used resource offerings are stored in the ClosedList. The Blackboard is divided into several regions and each region represents a subtask of a workflow. Which and how many regions exist depends on the workflow.

We give a simple example to outline the previous descriptions. Imagine, a user wants to store a video online. He also wants to convert the video from AVI to FLV and compress it to save disk space. The user normally does not care, how this workflow is completed in detail, he just wants a good, quick and cheap solution. Furthermore, the user does not want to start each subtask manually. He just defines the tasks and hands the video over to the Blackboard. Figure 1 gives a graphical representation.

The first question which shows up is: What is a good solution for the user? It is necessary to assign a numerical value to each subtask to make a comparison possible. These values are called "cost". Cost are constituted of "Quality of

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

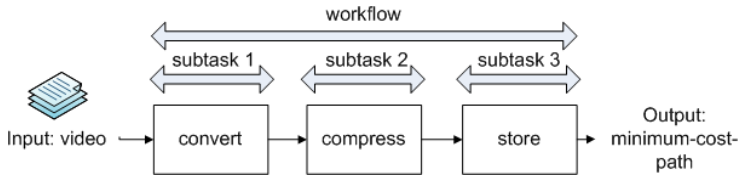


**Fig. 1.** Example of a workflow

Service" parameters, short QoS. The second question is: How can a user specify restrictions on the workflow? In our example, the user may need a minimum of 15GB of disk space to store the video, so it only makes sense to search for services which provide more than 15 gigabyte space. These rules are input parameters to the Blackboard. In our example we have the restriction, that the value for converting the video should be smaller than 60, the compression-ratio has to be greater than 20, the disk space has to be greater than 15.

Figure 2 shows a simple Blackboard consisting of 3 regions: convert, compress and store. For region "convert", three different services are available. To all services, a value is assigned. In this example we assume that there is only one cost-parameter for each service.

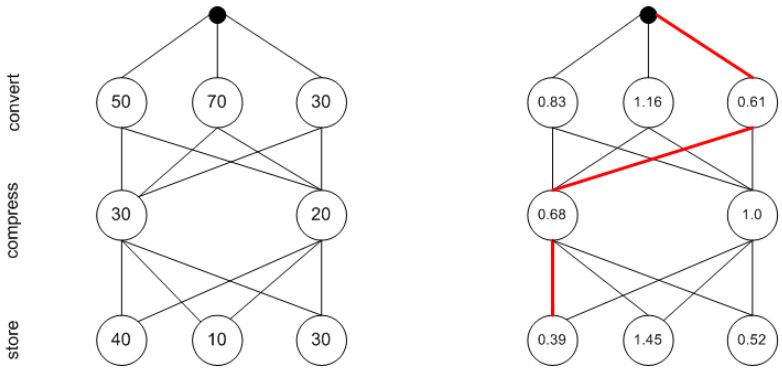


**Fig. 2.** Minimum cost path

We now have to find services which fulfill our restrictions, in other words, for the first region we search all services with an offer less than 60 and calculate the cost for each service. Then we look for the service with the minimal cost of each region. These are our optimal services.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

### 2.1 Finding the Best Service Provider

In practice, there is more than one value to describe the cost of a service. For example, the user is not only interested which services offer a disk space greater than 15GB, but he also includes the price for the storage in his calculations. To handle this scenario, we list all parameters of all providers and connect them, the result is a graph. Figure 3 shows the subtask "convert" and two providers (with ID 10 and 20). The first provider offers the conversion to AVI, the second offers the conversion to FLV and gives two more options to choose from, a faster and more expensive possibility (runtime) and a slower option with a lower price. Again, the user sets restrictions; in this example he chose FLV as output format, a runtime less than 80 and a price less than 60. Applying the formulas above to our parameters, we get the cost for each node. Note, that in case of Boolean-parameters, the cost are set to zero if the condition is true and set to infinity, if the condition is false.

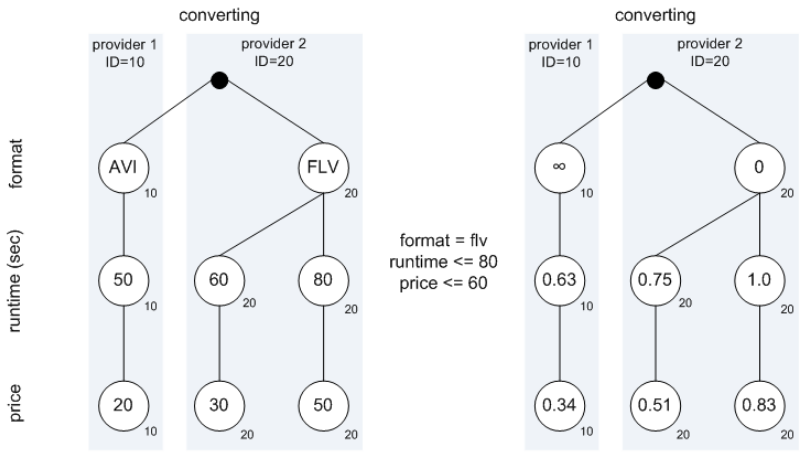


**Fig. 3.** Calculation of the cost of each node

To find the best provider, we use an algorithm to find the lowest cost path. The Dijkstra algorithm is an appropriate choice for this purpose. In some cases, the estimated cost for a node may be known in advance. If a good estimator is known and the risk of overestimating the cost is low, we can use the A-* algorithm instead of Dijkstra's algorithm to calculate the total cost.

### 2.2 Blackboard meets Dijkstra

Our Blackboard approach using Dijkstra's algorithm is defined by algorithm 1. The algorithm receives a list of regions (e.g. format, price and runtime) and a list of parameters as an input. Parameters are numerical values, such as

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

price=30. Each parameter is assigned to a service provider (serviceID), for example [price=30, serviceID=10].

```
input: List of regions, list of parameters
OpenList = [];
ClosedList = [];
path = [];
firstRegion = regionlist[0];
lastRegion = regionlist[len(regionlist)-1];
OpenList.add(allNodesOf(firstRegion));
retrace(Node)
    path += Node;
    if Node.region is firstregion then
        return
    else
        retrace(Node.ancestor)
    end
end
while OpenList not emtpy do
    calculateCosts(OpenList);
    currentNode = minimum_cost(OpenList);
    if currentNode.region = lastRegion then
        retrace(current);
    end
    foreach parameter in parameterlist do
        nextRegion = regionlist.index(current.region)+1;
        if parameter.region = nextRegion parameter not in OpenList and
          parameter not in ClosedList and parameter.serviceID =
          current.serviceID then
            parameter.cost = current.cost + parameter.cost;
            OpenList += parameter;
            parameter.ancestor = current;
        end
    end
    OpenList = OpenList \ current;
    ClosedList += current;
end
```

**Algorithm 1:** Find the best provider combintation

The approach is depicted in algorithm 1. It starts with the function calculateCosts() to determine the cost of each parameter. Then, the node with the lowest cost is added to the OpenList. The OpenList contains all known but not yet visited nodes. Then, it is checked if the current node is an "endnode" (which means, the last region of the board is reached). If this is not the case, all nodes

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

from the next region, which are provided from the same service provider as the current node are added to the OpenList. Finally, the current node is removed from the OpenList and added to the ClosedList. The ClosedList contains all nodes, which are already visited. The algorithm is running till there are nodes in the OpenList or the exit condition is fulfilled.

## 3 Workflow Optimization Implementation Architecture

Computational science applications all over the world can benefit from our framework. A real-world application scenario, which we used in the past [6], is the scientific workflow environment of the ATLAS Experiment aiming at discovering new physics at the Large Hadron Collider [1]. The TAG system [3] is a distributed system composed of databases and several web services accessing them.

To describe the implementation architecture of our Blackboard based optimization framework we use the Model-View-Controller concept. A URL opened in a standard Webbrowser represents the "view". The user has the opportunity to set rules via HTML-form, all rules are stored in a shared repository to which only the user and the Blackboard have access. The GUI also displays the results of the algorithm, i.e. those service providers with the best offer. Subsequently, the workflow executes autonomously: The user receives a response with the best offer and has to confirm it. Afterwards, all tasks of the workflow are completed automatically.

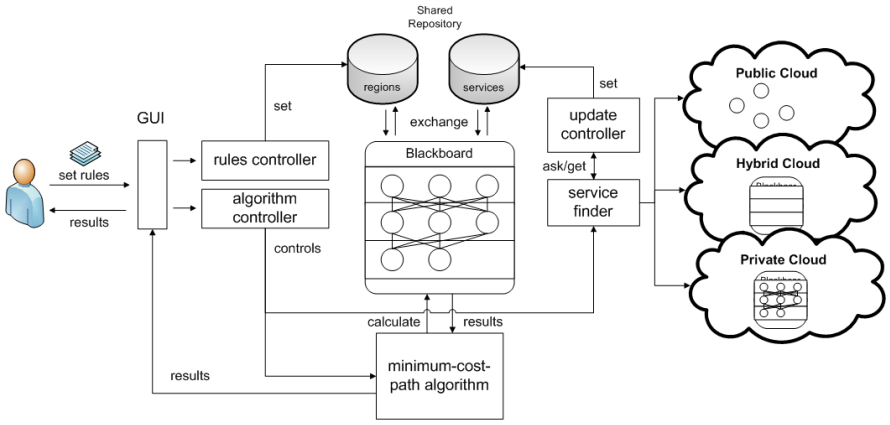


**Fig. 4.** The components of the sample implementation of the Blackboard approach

Figure 4 shows the graphical representation of the architecture: The left part shows the layer of the user, the right part is the service-layer. The core consists of the Blackboard, which is implemented in the sample-code as a class and

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38

several controllers that govern the process. In addition, a shared repository is used for services and rules as a cache, so that they can be accessed quickly by the Blackboard.

For justification the algorithm was implemented within the Google App Engine (GAE) [4]. The Blackboard-Application architecture of the software follows the concept shown in figure 4.

## 4 Conclusion

This paper presents an approach how the selection of services for scientific workflows on the Web can be optimized based on QoS offers of service providers. The novelty of our approach is mapping the service selection problem to a graph representation and to apply a combination of Dijkstra's algorithm and Blackboard method for optimization.

Further we present the architectural framework of the Blackboard approach and its components. The practical feasibility is shown by a use case implementation.

A further research topic is the handling of dynamically changing QoS conditions during workflow execution. The presented method can cope with this challenging situation, which is described in an extended version of this paper [7].

## References

1. Aad, G., Abat, E., Abdallah, J., Abdelalim, A., Abdesselam, A., Abdinov, O., Abi, B., Abolins, M., Abramowicz, H., Acerbi, E., et al.: The atlas experiment at the cern large hadron collider. Journal of Instrumentation **3**(8), S08003–S08003 (2008)
2. Corkill, D.D.: Blackboard systems. AI expert **6**(9), 40–47 (1991)
3. Duckeck, G., Jones, R.W.: Atlas computing. Technical design report by atlas collaboration, CERN (2005)
4. Engine, G.A.: https://developers.google.com/appengine/. last visited 07-04-2018
5. Vinek, E., Beran, P.P., Schikuta, E.: Classification and composition of qos attributes in distributed, heterogeneous systems. In: 11th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2011). IEEE Computer Society Press, Newport Beach, CA, USA (May 2011)
6. Vinek, E., Beran, P.P., Schikuta, E.: A dynamic multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment. Procedia Computer Science **4**, 166–175 (2011)
7. Vorhemus, C., Schikuta, E.: Blackboard meets dijkstra for optimization of web service workflows. arXiv preprint arXiv:1801.00322 (2017)
8. Wanek, H., Schikuta, E.: Using blackboards to optimize grid workflows with respect to quality constraints. In: Fifth International Conference on Grid and Cooperative Computing Workshops (GCC'06). vol. 0, p. 290–295. IEEE Computer Society, Los Alamitos, CA, USA (2006)
9. Yu, T., Lin, K.J.: Service selection algorithms for composing complex services with multiple qos constraints. In: ICSOC. vol. 3826, pp. 130–143. Springer (2005)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93713-7_38