EEM+08] Julian Eckert, Deniz Ertogrul, André Miede, Nicolas Repp, Ralf Steinmetz; **Resource Planning Heuristics for Service-oriented Workflows.** In: IEEE/WIC/ACM International Conference on Web Intelligence 2008, Sydney, Australia, December 2008. Seite 591–597

2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Resource Planning Heuristics for Service-oriented Workflows

Julian Eckert, Deniz Ertogrul, André Miede, Nicolas Repp, Ralf Steinmetz
Technische Universität Darmstadt
Multimedia Communications Lab - KOM
Merckstr. 25, 64283 Darmstadt, Germany
Julian.Eckert@kom.tu-darmstadt.de

Abstract

Resource allocation and resource planning, especially in a SOA and Grid environment, become crucial. Particularly, in an environment with a huge number of workflow consumers requesting a decentralized cross-organizational workflow, performance evaluation and execution-management of service-oriented workflows gain in importance. The need for an effective and efficient workflow management forces enterprises to use intelligent optimization models and heuristics to compose workflows out of several services under real-time conditions.

This paper introduces the required architecture Workflow Performance Extension - WPX.KOM for resource planning and workload prediction purposes. Furthermore, optimization approaches and a high-performance heuristic solving the addressed resource planning problem with low computational overhead are presented.

1. Introduction

The general outcome of the industrialization and consolidation in many industries leads to a dramatically increasing demand for agile and flexible business processes and business models. Furthermore, competitive markets intensify requirements in terms of cost-efficiency and performance characteristics of processes. Enterprises are forced to allocate and manage modular software artifacts at runtime. In this context a certain Quality of Service (QoS) level as well as an effective business process management has to be established in order to meet customer requirements.

Especially in the case of cross-organizational workflows, when services from internal as well as external partners are invoked in a single workflow, effective business process management is essential for reliable operations. The on-demand integration of external, loosely coupled services as well as the integration of internal legacy systems is provided by a Service-oriented Architecture (SOA) [15]. The realization of service-oriented workflows by loosely coupled, interoperable software artifacts (services) aims at empowering the invocation of distributed components in an event-driven or asynchronous fashion reflecting the underlying business process needs [16]. Thus, SOA is regarded as an approach to facilitate the needed flexibility and feasibility in aligning application landscapes to business-driven demands [13].

Accordingly, by allowing just-in-time integration and interoperability of multiple and alternative services (with the same functional properties), service-oriented workflows comprehend yet unsolved complexity and dynamics in resource allocation. Thus, in correspondence with the technological advances in internet technology, formerly long-term and thus more strategic decisions of resource selection develop strong operational character. In order to avoid the risk of poor workflow performance and the violation of Service Level Agreements (SLA), business process intelligence, resource planning of workflows, and performance analysis are equally important.

This paper addresses the problem of online resource allocation on the side of an orchestrator of service-oriented workflows under real-time conditions. To encounter the interfering complexity of service-allocation and -utilization, we assume a multistage concept of first allocating the most appropriate services and hereupon scheduling service utilization to be appropriate. Thereby we focus on discussing the selection of Web services as an optimization problem, regarding user's preferences and constraints on the non-functional attributes of the services. Both, allocation and corresponding utilization of services have to aim at maximizing a defined utility function under customer-defined overall QoS constraints on the resulting workflow [12].

However, QoS-aware Web service composition with regard to the overall QoS requirements and resource constraints, especially under real-time conditions, has no satisfactory solution yet. Thus, this document aims at depicting a heuristic, feasible for solving the resource planning problem under real-time conditions.

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.108

The remainder of this paper is structured as follows. After an overview about related work in section 2 the considered scenario is described in section 3. The required architecture as well as the detailed resource planning approach are depicted in section 4 and 5, respectively. After an evaluation in section 6, the paper closes with a summary and an outlook on future work.

2. Related work

Composition and modeling of functional characteristics of service-oriented workflows has received much interest. Mostly semantic [6] [19] or rule-based [7] [18] approaches have been proposed for this problem. However, non-functional QoS issues, especially that of the resulting composite workflow, have been addressed by these approaches only partially.

Research in the environment of dynamic Web service composition recently concerns the mentioned overall QoS behavior of the entire workflow. These approaches aim at optimization of multiple QoS-criteria (such as price, response time, and reliability) while regarding global constraints on the composite workflow. This combinatorial optimization problem is proven to be NP-Hard [12] [14] [17].

Graph-based approaches map the composition problem to a Multi-Constrained Optimal Path (MCOP) problem. Lin and Yu present a heuristic for the Multi-Constrained Shortest Path (MCSP) problem with polynomial complexity [12]. However, this is, in its concrete runtime-behavior, still not suitable for real-time application. When modeling a network impact (QoS-value of the network linkage), the graph-model's comparative suitability increases with the relative weight of network conditions. Moreover, graph-based models can solve multiple process plans at the same time (e. g. alternative workflow routes).

Combinatorial approaches transform the composition problem into a Multi-choice Multi-dimension Knapsack Problem (MMKP) [14]. Aggarwal et al. and Zeng et al. both propose approaches using the integer linear programming method to exactly solve the service selection problem as an instance of MMKP [1] [21]. However, this is too complex for run time decisions [12] [20].

Meta-heuristics have been proposed by several authors for solving the MMKP-problem. Canfora et al. and Khuri et al. discuss genetic algorithms and state that these perform much better and reveal significant improvement in terms of scalability compared to conventional linear integer programming approaches [5] [11]. The authors confirm outperformance in terms of runtime-behavior and scalability compared to linear programming. However, they reveal marginal improvement of the solution with drastic increment in computation time, when using the solution of their proposed heuristic as an initial solution. Akbar et al. analyze the weak worst-case complexity of these approaches compared to efficient greedy heuristics such as [2] [10].

Besides these considerations there also have been attempts to optimize the worst-case and the average-case performance behavior of distributed service-oriented workflows [8] [9]. However, none of these optimization approaches has considered limited service execution capacities yet.

3. Scenario

The scenario of this work focuses on service-oriented cross-organizational workflows. Assuming a large set of workflow consumers, requesting a specific workflow at a specific time period, the question arises if it is possible to offer all of them a workflow execution at demanded performance properties. This implies that an arbitrary workflow orchestrator has to be able to determine optimal service compositions for all workflow consumers. As described in section 2, several approaches exist to determine optimal execution plans for *one* specific workflow execution request.

In contrast, the considered scenario includes multiple workflow consumers, requesting workflow executions at the same time (Figure 1). Such workflows can invoke internal as well as external services from other enterprises. Thus, in order to be able to execute a huge amount of simultaneous execution requests, an intermediary must apply request managing.

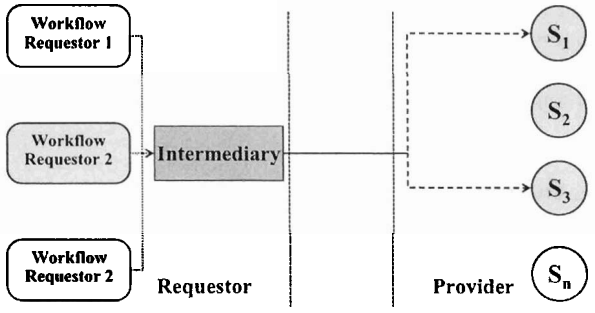

Figure 1. Research scenario

The intermediary has to monitor, collect, and prioritize all workflow execution requests. Moreover, he is responsible for the detailed resource planning, i. e. the selection and invocation of services under several constraints. Assuming several priority classes such as gold, silver, and bronze requestors, optional execution plans with several QoS levels have to be specified. For huge amounts of incoming execution requests, effective resource planning is essential in order to avoid performance degradation. Multiple services may have to be invoked in parallel. The challenge will be to determine cost-efficient execution plans that fulfill the QoS

demands and guarantee the execution of all workflow execution requests.

4. WPX.KOM – An architecture for resource planning

The following resource planning architecture represents an architectural extension to the proxy architecture WSQoSX.KOM [3]. This architecture for managing arbitrary Web service workflows is able to compose workflows from multiple services while considering multiple QoS demands, such as response time, cost, etc. The modular architecture consists of several components as an Accounting, QoS Monitoring, and SLA Management component.

From QoS perspective, the most important component in this architecture is the Service Selection Component. Service selection models and heuristics for QoS aware optimization of single workflow requests are well studied in literature, e. g. [4]. However, these architectures do not consider multiple parallel workflow execution requests and parallel service invocations.

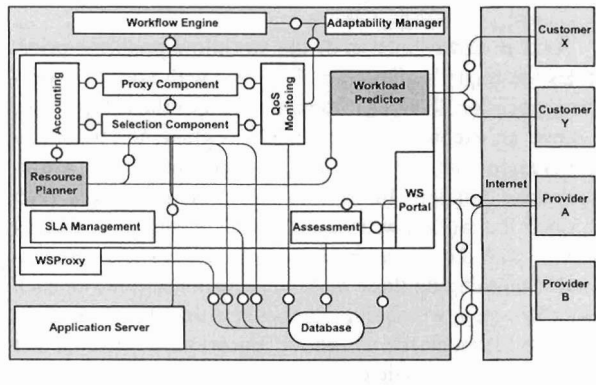

Figure 2. Workflow Performance Extension – WPX.KOM.

The presented research aims at service composition for a large amount of workflow requests and the associated problem of limited service capacities. The question is which components are necessary to handle a large amount of incoming requests from different priority classes with several QoS requirements. To overcome these challenges, the Workflow Performance Extension - WPX.KOM has been developed. Mandatory architectural components, such as a *Resource Planner* and a *Workload Predictor* are included in this architecture. The *Workload Predictor* has to monitor incoming workflow execution requests and makes forecasts for future workload. The *Resource Planner* has to determine optimal service compositions, check whether the execution capacity of any service is reached, replace services by other services and specify which services have to be invoked in parallel in order to be able to serve all incoming requests. The tasks of the two architectural components are:

- Workload Predictor
  - Monitoring of incoming workflow execution req.
  - Quantitative forecasting
  - Request Prioritization
- Resource Planner
  - Determination of efficient service composition
  - Checking of service execution capacity
  - Replacing of services
  - Execution scheduling based on workload predict.

The *Resource Planner* receives its information from the *Workload Predictor* and has to communicate with the Accounting and the QoS Monitoring component. For resource selection, the *Resource Planner* has to gather information about costs and QoS properties of invoked services to build up an efficient execution plan for the next planning period. The detailed procedure of this approach is shown in the following section.

5. Resource Planning

This section comprises an overview of the developed optimization model that maps the above described resource planning problem. Moreover, it provides a feasible service selection procedure for the *Resource Planner*, based on a developed high-performance heuristic.

5.1. Optimization model

The depicted resource planning problem can be described as follows. A composite workflow has to be determined, by selecting (a) specific service(s), out of a multitude of available services with the same functional parameters for each workflow-step. At this, the decisions in each workflow-step are interconnected by global constraints on the resulting workflow. In each set of functionally equivalent services, the alternatives are solely distinguished by their non-functional properties (such as response time, execution capacity, costs etc.). Thus, the objective of a selection is to maximize a defined non-functional utility of the composite workflow (which is a result of the respective properties of the utilized services), regarding defined constraints on the entire workflow. This combinatorial optimization problem can be mapped to a NP-hard multiple-choice multi-dimensional knapsack problem (MMKP) [14].

In the following, there will be a presentation of a specific instance of this optimization model. In the presented model,

the objective of the selection will be to minimize the service invocation costs, regarding a restriction on the overall response time of the workflow. In accordance, there is only a one-dimensional constraint observation (response time). A specific overall response time for a specific maximum bulk of requests must not be violated, whilst the selection must comprise enough processing capacity in each workflow step for this specific maximum bulk of requests.

As we analyze workflows with a sequential order of process steps, a workflow consists of n tasks. Task $i(i = 1, ..., n)$ has to be executed before task $i'(i' = 1, ..., n)$ if $i < i'$. The set of m_i different services that provide the required functionality for task i is called category i. If the binary variable $x_{i,j} = 1$, it means that Web service j of category i is selected to be executed within the execution plan. Executing $WS_{i,j}$ within the workflow means that the workflow-engine requests the respective service supplier in a single request to deliver $k(k = 1, ..., |cap_{i,j}|)$ units of Web service executions. Subsequently the service provider processes the request and returns after $t_{i,j}$ time units. Thus, Web service capacity is a discrete portion of parallel executions.

The following list gives an overview about indices and parameters used in the following consideration:

$i = 1, ..., n$	: Index for workflow-step
$j = 1, ..., m_i$	: Index for candidate services in category i
$c_{i,j}$	: Costs caused by $WS_{i,j}$
$t_{i,j}$	: Response-time of $WS_{i,j}$
$cap_{i,j}$	: Processing capacity of $WS_{i,j}$
$\tau > 0$	: Workflow response time restriction
$W > 0$	: Workload

Here, i denotes the workflow-step and j a specific Web service of its service category. The cost accounted for the execution of each service $WS_{i,j}$ is denoted as $c_{i,j}$, whilst $t_{i,j}$ denotes its response time and $cap_{i,j}$ its specific execution capacity. The service providers offer their services according to the following conditions: Up to its maximum execution capacity $cap_{i,j}$, a service $WS_{i,j}$ always reveals a fix response time $t_{i,j}$ and fix costs $c_{i,j}$ independent from the workload. This implies that the specific service can handle up to $cap_{i,j}$ service executions in parallel with an equal response time. Except from practical issues, these assumptions reflect a worst-case observation in order to be able to guarantee a certain QoS.

Equation (1) shows the binary variable $x_{i,j}$ that describes whether the corresponding Web service $WS_{i,j}$ is selected in the respective composition or not:

$$x_{i,j} \in 0, 1 \tag{1}$$

Objective funtion:

$$MinF(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} x_{i,j} * c_{i,j} \tag{2}$$

Constraints:

$$\sum_{j=1}^{m_i} x_{i,j} * cap_{i,j} \geq W \qquad \forall i = 1, .., n \tag{3}$$

$$\sum_{i=1}^{n} max\left\{t_{i,j} | x_{i,j} = 1\right\} \leq \tau \tag{4}$$

The objective function depicted in equation (2) assures the selection of a set of services at minimal overall costs. Equation (3) ensures for all process steps $(i = 1, ..., n)$ a selection of a set of services that is able to handle all incoming execution requests W. This can either be achieved by choosing one specific Web service with a potential execution capacity exceeding the amount of execution requests, i. e. $cap_{i,j} \geq W$ or a set of services whose sum of processing capacities exceeds the requested execution capacity, e. g. $cap_{1,1} + cap_{1,8} \geq W$.

As depicted in section 4, the workflow predictor prioritizes the requests into several classes with specific workflow execution deadlines. Regarding a specific category we assume an execution deadline τ. To guarantee workflow termination within τ time units, we have to make a bottleneck consideration. In order to determine the longest path through the workflow, in each workflow-step the selected service with the maximum response time has to be considered. Summing up these maximum response times in each workflow-step, we obtain the response time for passing all workload through the workflow. Equation (4) ensures that an execution plan is determined that holds the deadline τ.

We have seen that due to its combinational nature, the MMKP is proven to be NP-Hard [12]. Moreover, the combinatorial complexity of the model presented here, exceeds that of the MMKP. Within each workflow-step, not only one, but multiple services can be chosen. However, our optimization model can be reduced to the MMKP.

In order to do so, it is necessary to determine all valid subsets (satisfying capacity requirements AND no service can be removed without violating capacity requirements) of the services in each category i. Assuming a finite set S of services with cardinality $|m_i|$ in each category i, then the power set of S has cardinality of 2^m. The power set represents the set of all subsets of S. In the worst-case it is necessary to enumerate each subset. By enumerating and representing each valid subset as a single service with aggregated properties of the included services, we can map the presented model to the NP-hard MMKP. In the worst-case this means a computational overhead co in relation to [12]

of $co = n * 2^m$. In accordance, the presented optimization problem is also NP-hard and thus, requires an efficient heuristic as described in the next subsection.

5.2. Heuristic H1_G.KOM

To overcome the exponential computational complexity of this combinatorial optimization problem, this section provides an efficient heuristic for the adressed resource planning that can be applied to problems with multi-dimensional constraints on the entire workflow.

Its general approach is to find a near-optimal solution outgoing from an initial feasible solution. Outgoing from an initial feasible solution, in case of multi-dimensional constraints on the workflow, a single-dimension reduction is applied. In the next step, analogue to [20], the achieved items are sorted in a single list according to a metric referring to the utility relative to the resource consumption [2]. Ultimately, using the sorted list, an approximation algorithm is applied in order to reach a near-optimal feasible solution. A pseudo code of the proposed heuristic can be seen in the following.

```
forall s_{i,j} in ascending order
  currentSelection_i = initially selected services in category i
  newSelection_i = {}
  newSelection_i = newSelection_i ∪ X_{i,j}
  k = 0
  While (capacity of newSelection_i < W) ∧ (k ≤ m_i)
    if (X_{i,j} from l_i [k] ∉ newSelection_i)
      newSelection_i = newSelection_i ∪ X_{i,j}
      replan list l_i according to remaining workload
    k++
  if capacity of newSelection_i ≥ W
    Exchange currentSelection_i with newSelection_i
    if ¬(constrSatisfied) ∨ (newCost > oldCost)
      degrade random previous selection
      recalculate newSelection_i and retry insert
      if ¬(constrSatisfied) ∨ (newCost > oldCost)
        reinsert currentSelection_i
```

Applying H1_G.KOM, the *first step* is to determine a feasible initial solution. As we assume a positive correlation of a Web service's costs $c_{i,j}$ to its' execution capacity $cap_{i,j}$ and a negative correlation to its' response time $t_{i,j}$, more expensive services are more likely to satisfy constraints on the given resources. Thus, H1_G.KOM generates the initial solution by selecting the most expensive service(s) in each category, starting with $WS_{i,\alpha}$ ($c_{i,\alpha} = max\{c_{i,j}|j = 1,..,m_i\}$) to find a feasible solution. For complexity evaluation, in the following we assume all $m_i = m$. Thus, the worst-case complexity of this step (no feasible solution exists) is $O(z * m * n)$ (with z = number of resources).

In a *second step*, in case of multi-dimensional constraints on the workflow, a dimension reduction is performed. Assuming z resources, H1_G.KOM calculates the consumption of each resource $k(k = 1,..,z)$ by the entire workflow of the initial solution $R\alpha_k = \sum_{i=1}^{n} r_{i,y,k}$, $(x_{i,y} = 1)$. By dividing the effective resource consumption of each resource k through its constraint value W_k, we achieve a penalty vector. The penalty $p_k = R\alpha_k / W_k$ for resource k is proportional to its utilization in the initial solution and thus, indicates its relative expensiveness. These penalty weights are then used to calculate for each $WS_{i,j}$ the consumption of the aggregate single resource $R_{i,j} = \sum_{k=1}^{z} p_k * r_{i,j,k}$. This step is not necessary to solve the presented model, as we regard a one-dimensional constraint on the entire workflow (response time). The complexity of this step is $O(n * (z + m))$.

The *third step* is to generate a single priority list of candidates to be part of the workflow. The aim is to sort candidates according to a metric depicting their utility relative to their resource consumption. The achieved two-dimensional search space in the presented model is cost and time consumption. As maximizing the utility of a selection corresponds to minimizing its costs, we calculate for each alternative $WS_{i,j}$, a metric $s_{i,j}$ as the costs multiplied with the resource consumption, as we want to minimize both. Additionally, this value is divided by the relative offered execution capacity as less capacity infers less utility and thus, more costs. This parameter represents the potential cost that service $WS_{i,j}$ would induce to satisfy the demanded workload in workflow-step i (in the presented model $R_{i,j} \hat{=} t_{i,j}$).

$$s_{i,j} = \frac{c_{i,j} * R_{i,j}}{\boldsymbol{min}\left\{1, \frac{\boldsymbol{cap_{i,j}}}{\boldsymbol{W}}\right\}} \tag{5}$$

The points sorted in a single list, build a priority list of candidates to switch with services in the initial solution. The complexity of this step is $O(n * m + n * log(n))$.

In a *fourth step*, the heuristic applies a switching algorithm. Going in ascending order through the sorted list of $s_{i,j}$ values, it tries to exchange the respective Web services with those that are currently selected in the workflow. This will incrementally enhance the solution by gradually saving costs and approaching up to the resource constraints. As not only single Web services, but sets of services have be selected in each category to satisfy workload, the service corresponding to $s_{i,j}$ may be amended with other services from category i. In order to reduce the complexity of this operation, H1_G.KOM additionally adds all candidates in *step three* into priority lists $[l_i]$ for each workflow-step i. Moreover, these lists allow dynamic reordering of candidates according to the remaining workload. For this purpose, the metric in each list $[l_i]$ is defined as

$$l_{i,j} = \frac{c_{i,j} * R_{i,j}}{min\left\{1, \frac{cap_{i,j}}{W - \sum_{j=1}^{m_i} cap_{i,j} * x_{i,j}}\right\}} \tag{6}$$

Another feature of the heuristic is, that when going through the global list of $s_{i,j}$ values, it can revise previously taken decisions to avoid local optima. If exchanging of a service selection fails due to lower utility or constraint violation, it retries an exchange by degrading a randomly picked previous selection. The worst-case complexity of this step is $O(n * m^2)$.

6. Evaluation

This section provides an evaluation of the developed heuristic H1_G.KOM. In reference to the exact solution of an integer programming approach, it will evaluate computation performance and solution quality on varying problem sizes. The experiments were run on an Intel Centrino (2 GHz) system with 3GB of RAM, running Windows XP. We implemented a data generator and a simulation engine with the Java JDK version 1.6.0.06.

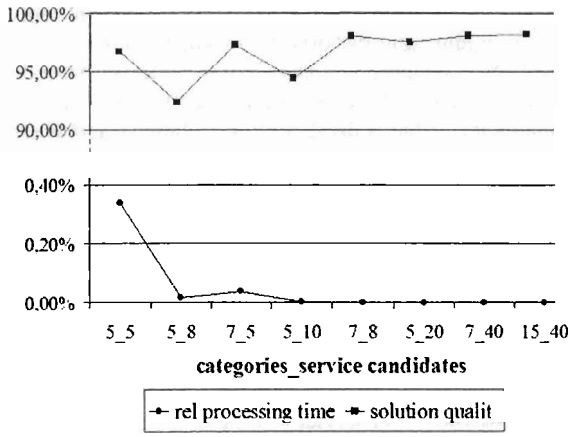

Figure 3. Impact of problem size

The data generator creates test cases with specific numbers of categories and category sizes. Our evaluation analyzes the impact of 1. the problem size (number of categories and service candidates) and 2. the strength of resource constraints, by comparing the results of H1_G.KOM to those of the exact approach. For each set, 20 test cases were processed by H1_G.KOM and a linear integer program in lp_solver [1] (using a simplex algorithm and Branch&Bound).

The created sets vary in the number of categories and service candidates ($m_i = m \, \forall \, categories$) as shown in Figure 1. For the service properties we defined a random variable C for costs with a dispersion about 60% around its mean value ($E(C)$). Further variables T for response time (correlation to costs about -0.9) and CAP for capacity (+0.9) were defined. In each test case, the restriction of τ was set to 90% of the potential average response time of the workflow ($0.9 * n * E(T)$) and W was set to 25% of the potential average execution capacity in a category ($0.25 * m * E(CAP)$).

[1] http://lpsolve.sourceforge.net/5.5/, Version 5.5.0.12

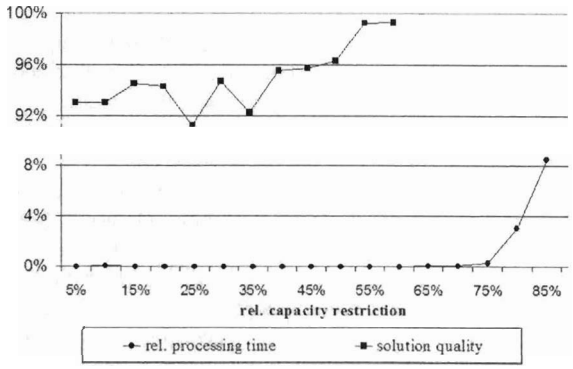

Figure 4. Impact of capacity restriction

With a relative processing time less than 0.4% (vastly decreasing with increasing problem size), the heuristic significantly outperforms the integer programming approach, whilst solution quality fluctuates about 95%. In absolute values, processing times of H1_G.KOM remain on an extremely low level, feasible for runtime conditions as shown in Table 1 and 2. Even larger problems about 7 categories and 40 candidates are solved in less than one second.

cat. _ service cand.	5_5	7_5	5_8
lp_solver (ms)	476.72	6766.86	27106.89
H1_G.KOM (ms)	1.72	1.62	3.92

Table 1. Computation time 1

Regarding capacity constraints, the results reveal most computational load about 35% to 45% workload demand. With increasing workload, it is less likely to find a feasible solution and thus, processing times vastly decrease.

cat. _ service cand.	5_10	7_8	5_20
lp_solver (ms)	595264.53	$\approx$ 38000000	$\approx 7^{20}$
H1_G.KOM (ms)	6.94	5.67	71.82

Table 2. Computation time 2

However the evaluation shows that the developed heuristic H1_G.KOM provides very good results in terms of solution quality and overall computational performance. Without this developed heuristic it would not be possible for an intermediary to plan all services in terms of an efficient resource planning in real-time.

7. Conclusion

Resource planning of service-oriented workflows becomes increasingly important, considering limited service execution capacities. The addressed resource allocation problem is extremely complex, $(2^m)^n$ possible compositions. In accordance, this paper introduces the needed Workflow Performance Extension – WPX.KOM as well as an optimization model. Concerning real-time requirements it is not possible to compute an optimal solution. Instead, the proposed heuristic H1_G.KOM achieves near-optimal solutions at a good performance as extensive evaluation reveals. H1_G.KOM outperforms any exact integer programming approach, whilst achieving very good solution approximation; both converging to the better with increasing problem size.

Our further research aims at optimizing and extending the developed heuristic and developing a holistic resource planning approach by including several cost models as well as workload prediction.

8. Acknowledgements

This work is supported in part by E-Finance Lab Frankfurt am Main e.V. (http://www.efinancelab.com).

References

[1] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint Driven Web Service Composition in METEOR-S. In *IEEE International Conference on Services Computing (ICSOC 2004)*, pages 23–30, 2004.

[2] M. M. Akbar, M. S. Rahman, M. Kaykobad, E. G. Manning, and G. C. Shoja. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. *Computers and Operations Research*, 33(5):1259–1273, 2006.

[3] R. Berbner, O. Heckmann, and R. Steinmetz. An Architecture for a QoS driven composition of Web Service based Workflows. In *Networking and Electronic Commerce Research Conference (NAEC 2005)*, 2005.

[4] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. An Approach for Replanning of Web Service Workflows. In *12th Americas Conference on Information Systems (AMCIS 2006)*, 2006.

[5] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 1069 –1075, 2005.

[6] J. Cardoso. *Quality of Service and Semantic Composition of Workflows*. PhD thesis, Department of Computer Science, University of Georgia, 2002.

[7] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan. Adaptive and dynamic service composition in eFlow. Technical Report HPL-200039, Technology Laboratory, Palo Alto, CA, March 2000.

[8] J. Eckert, S. Schulte, M. Niemann, N. Repp, and R. Steinmetz. Worst-Case Workflow Performance Optimization. In *3rd International Conference on Internet and Web Applications and Services (ICIW 2008)*, 2008.

[9] J. Eckert, S. Schulte, N. Repp, R. Berbner, and R. Steinmetz. Queuing-based Capacity Planning Approach for Web Service Workflows Using Optimization Algorithms. In *IEEE International Conference on Digital Ecosystems and Technologies (DEST 2008)*, 2008.

[10] S. Khan, K. F. Li, E. G. Manning, and M. D. M. Akbar. Solving the Knapsack Problem for Adaptive Multimedia Systems. *Studia Informatica Universalis*, 2(1):157–178, 2002.

[11] S. Khuri, T. Bäck, and J. Heitkötter. The Zero/One Multiple Knapsack Problem and Genetic Algorithms. In *Symposium on Applied Computing (SAC 1994)*, pages 188–193, 1994.

[12] K.-J. Lin and T. Yu. Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. In *3rd International Conference on Service-oriented Computing (ICSOC 2005)*, 2005.

[13] Z. Mahmood. Service Oriented Architecture: Potential Benefits and Challenges. In *11th WSEAS International Conference on Computers (ICCOMP 2007)*, pages 497–501, 2007.

[14] S. Martello and P. Toth. Algorithms for Knapsack Problems. *Annals of Discrete Mathematics*, 31:70–79, 1987.

[15] M. P. Papazoglou. Service-Oriented Computing: Concepts, Characteristics and Directions. In *4th International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–12, December 2003.

[16] M. P. Papazoglou and W.-J. van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal –The International Journal on Very Large Data Bases*, 16(3):389–415, 2007.

[17] R. Parra-Hernandez and N. J. Dimopoulos. A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(5):708–717, 2005.

[18] S. R. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Web Service Composition. In *11th International World Wide Web Conference (WWW 2002)*, 2002.

[19] M. Sheshagiri, M. desJardins, and T. Finin. A Planner for Composing Services Described in DAML-S. In *AAMAS Workshop on Web Services and Agent-based Engineering*, 2003.

[20] C. Ykman-Couvreur, V. Nollet, F. Catthoor, and H. Corporaal. Fast Multi-Dimension Multi-Choice Knapsack Heuristic for MP-SoC Run-Time Management. In *International Symposium on System-on-Chip (ISSOC 2006)*, pages 1–4, 2006.

[21] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *12th International World Wide Web Conference (WWW 2003)*, pages 411–421, 2003.