An Approach for Compliance-Aware Service Selection with Genetic Algorithms

Fatih Karatas[1] and Dogan Kesdogan[2]

[1] Chair for IT Security Management, University of Siegen, D-57072 Siegen, Germany
[2] Chair for Management Information Systems IV, University of Regensburg, D-93055 Regensburg, Germany
karatas@wiwi.uni-siegen.de, kesdogan@ur.de

Abstract. Genetic algorithms are popular for service selection as they deliver good results in short time. However, current approaches do not consider compliance rules for single tasks in a process model. To address this issue, we present an approach for compliance-aware service selection with genetic algorithms. Our approach employs the notion of compliance distance to detect and recover violations and can be integrated into existing genetic algorithms by means of a repair operation. As a proof-of-concept, we present a genetic algorithm incorporating our approach and compare it with related state-of-the-art genetic algorithms lacking this kind of check and recovery mechanism for compliance.

Keywords: Service-oriented Computing, Service Selection, Compliance, Multi-objective Optimization, Genetic Algorithms.

1 Introduction

Service-oriented computing (SOC) is a favored approach for developing distributed applications by orchestrating loosely coupled services according to a process model [1]. Each service realizes a well-defined task at a certain Quality-of-Service (QoS) level. In this regard, service selection algorithms are employed to determine service compositions for given process models. Mathematically this problem is usually represented as Multidimensional Multiple-choice Knapsack Problem (MMKP) [2]. As MMKP is NP-hard [3], heuristic approaches (e.g. genetic algorithms, GAs) are favored to find near-optimal solutions in short time.

Processes need to be compliant with rules originating from sources such as the *Health Insurance Portability and Accountability Act* (HIPAA) and ISO 27001. These issues are usually considered to be part of process definition and execution [4]. However, certain aspects of compliance such as location of execution can be compromised by heuristic service selection (see section 3.1). Proposed approaches for service selection based on GAs such as [5–7] do not address this issue.

This paper contributes the following: 1) To our best knowledge the first discussion which compliance aspects (see section 2) might be violated by heuristic service selection, 2) a method to detect and recover compliance violations in heuristically determined service compositions utilizing the notion of compliance

S. Basu et al. (Eds.): ICSOC 2013, LNCS 8274, pp. 465–473, 2013.

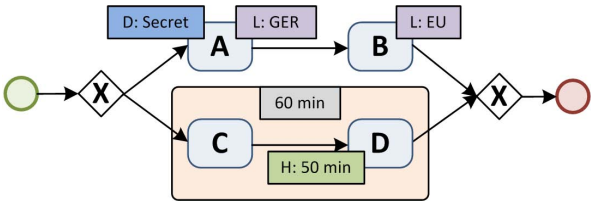


Fig. 1. Sample workflow with compliance annotations

distance as introduced by Sadiq et al. [8] and 3) a GA called COMPAGA based on NSGA-II which incorporates our method by means of a repair operation. This algorithm is tested against several state-of-the-art GAs (see section 4).

The remainder of this paper is structured as follows. Section 2 motivates the presented work. Our approach is presented in Section 3 and evaluated in Section 4. Section 5 covers related works. Finally, the paper is concluded in Section 6.

2 Motivation

Fig. 1 shows a sample process model which is the starting point for service composition. The aim of QoS-aware service selection is to determine compositions which fulfill the QoS constraints of a process model. While for small and medium size problems exact solutions can be determined, it is not feasible for real-world problems. GAs (see e.g. [5–7]) proved to deliver good results in short time.

Processes might also be subject to semantic constraints arising from e.g. standards and regulations. Semantic constraints are called *compliance rules* and can refer to one or more *compliance aspects* (see below). A process is called *compliant* if it does not violate any of its compliance rules [9]. In the following we will introduce the five compliance aspects differentiated in the literature (e.g. [9–11]), name the primary question which each addresses and cite and example from either HIPAA or the *German Federal Data Protection Act* (BDSG):

- **Activities:** Which tasks are performed in which sequence? (BDSG:4) requires user agreement to personal data collection prior to any such action.
- **Data:** What data objects are produced and consumed and which management rules are applied? (BDSG:3a) requires anonymization or psudonymization of personal data if there is no need to access such data in plain.
- **Location:** Where are tasks performed? (BDSG:4b) requires that personal data might be transferred to 3rd parties outside of Germany only if an appropriate level of protection can be assured by the 3rd party.
- **Resources:** By whom are tasks performed? (HIPAA:164.530.a.1.ii) requires that institutions implementing HIPAA must appoint a person or office responsible for receiving privacy complaints.
- **Time limits:** Within which time constraints are tasks being performed? (HIPAA:164.524.a.1; HIPAA:164.524.b.2.i) empowers patients to be informed about their stored protected health infomration (PHI) within 30 days.

3 Approach

In this section we first discuss which compliance aspects might be violated by heuristic service selection. Next we present our approach for detecting compliance violations and finally our algorithm for recovering compliance.

3.1 Impact of Compliance Aspects on Heuristic Service Selection

Activities: The sequence of tasks is defined by the process modeler. As service selection algorithms take process models as input to determine suiting compositions, this aspect is out of scope for service selection. A number of approaches exist to ensure the compliance of process models during design time (see section 5). Thus in the following we assume process models to be compliant regarding the sequence of activities when service selection is performed.

Data: Depending on the type of data, services might be required to fulfill a minimum of security measures (e.g. encryption strength). This leads to the necessity to consider these requirements as local constraints. Thus this aspect needs to be considered by service selection algorithms.

Location: Depending on the type of data, processing might be restricted to certain countries and/or regions. This usually applies to single tasks and thus needs to be considered as local constraint in service selection.

Resources: Ensuring that certain tasks are restricted to certain entities can be achieved e.g. utilizing credentials. This assignment of entities to tasks needs to be performed by the process modeler. Restricting invocation of services to authorized entities is mainly a question of configuration and thus out of scope.

Time limits: Processes might contain non-human as well as human tasks. For human tasks, process designers usually allot a certain amount of time for completion. Thus, service selection for sub-processes subject to time limits needs to consider the time limit as well as alloted times for human tasks.

3.2 Detecting Compliance Violations

Compliance of service compositions is measured utilizing compliance distance as introduced by Sadiq et al. [8]. Compliance distance is a quantitative measure which in its basic form counts the number of compliance violations in a process instance. Here it is adapted for service selection and counts the number of violations caused by selected services in a composition. This basic view assumes that consequences of compliance violations are equally bad. Sadiq at al. pointed out that a more sophisticated approach is to associate a cost with each violation and to define compliance distance as the sum of violation costs [8]. For the sake of simplicity we will use the basic measure in the following.

Data and *location* violations can be detected locally. *Time limit* scopes may be nested and be composed of human as well as non-human tasks. Thus a data structure *timeScopes* is defined containing one list *scope* per time scope in a

process model as well as its *time limit*. Given a process model P and a service composition SC, the compliance distance of SC can be determined in two steps. First, for each task $p \in P$ it is checked if the selected service in SC fulfills *data* and *location* requirements defined in P. If this is not the case, the index of p is stored in a list V. In case that p is part of one or several time scopes, the index of p is stored in all respective lists $scope \in timeScopes$. Secondly, all lists $scope \in timeScopes$ are iterated to sum up times allotted to human tasks as well as response times of selected services in SC. If a sum is greater than the *time limit* assigned to *scope*, all elements $p \in scope \;.\; p \notin V \wedge \neg p.isHumanTask$ are added to V. The compliance distance of SC then equals $|V|$. Algorithm 1 shows the pseudo-code for this operation.

3.3 Recovering Compliance

In order to replace services efficiently in logarithmic time, the set of service alternatives is clustered with three levels. The first level clusters services according to service class S_i, the second to *data* class and the third to *location* where the latter two are interchangeable (see fig. 2).

Given such a clustering, P, V, and a non-compliant SC, the repair operation works as follows. For each violation $v \in V$ the corresponding service class S_v as well as the set C of *data* and *location* constraints are determined. Then the clustering is searched for a service of class S_v which has a) at least the required

Algorithm 1. Detect Compliance Violations(P, SC)

```
1  V := empty list;
2  timedScopes := data structure with one list per time scope and its time limit;
3  foreach p ∈ P do
4  |  C := p.getConstraints;
5  |  foreach c ∈ C do
6  |  |  if c.isDataAnnotation ∨ c.isLocationAnnotation then
7  |  |  |  Check if SC[p] meets compliance requirement c;
8  |  |  |  if SC[p] does not meet c then add p to V
9  |  |  else if c.isTimeConstraint then
10 |  |  |  Add p to all corresponding lists in timedScopes;
11 |  |  end
12 |  end
13 end
14 foreach scope ∈ timedScopes do
15 |  if Σ human processing times + Σ response times > scope.timeLimit then
16 |  |  foreach p ∈ scope . p ∉ V ∧ ¬p.isHumanTask do
17 |  |  |  Add p to V;
18 |  |  end
19 |  end
20 end
21 return V;
```

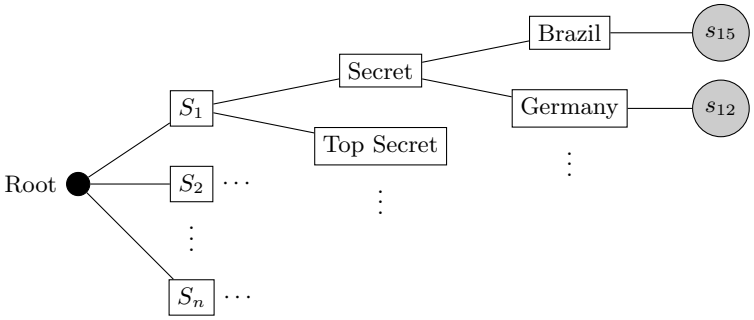


Fig. 2. Service clustering

data class, b) the required *location* and c) minimum response time. If such as candidate exists, it replaces the old service of class S_v in SC (see algorithm 2).

The success of the repair operation depends on the existence of suiting service alternatives. As such it cannot be guaranteed that a non-compliant SC becomes compliant afterwards. *Time limit* constraints are addressed implicitly by picking services with minimum response time. Explicit addressing would again require a two-step approach as in algorithm 1. Our approach will lead to a compliant SC in terms of *time limits* if suiting service alternatives exist. Otherwise *time limit* constraints cannot be met without violating another local constraint. Therefore a more sophisticated approach is not necessary.

To test our repair operation, we implemented a GA based on NSGA-II [12] called COMPliance-Aware GA (COMPAGA). COMPAGA first generates a random initial population and then iteratively performs selection, crossover and mutation operators on this population. Next it calculates the compliance distance of the new offspring with algorithm 1. If the compliance distance of the offspring is ≥ 1, the repair operation (see algorithm 2) is performed with a probability of p_{rep}. Experiments with different values for p_{rep} showed 75% to be a good compromise between runtime and average compliance distance of obtained solutions (see section 4.2).

Algorithm 2. Repair operation(P, SC, V)

```
1  Clustering := Three-level clustering of service alternatives;
2  foreach v ∈ V do
3  |  S_v := service class of v;
4  |  C := data and location constraints for S_v in P;
5  |  cand := Clustering.pick(s ∈ S_v . s.data ≥ C.data ∧ s.location = C.location
6  |                                  ∧ s.responseTime = min);
7  |  if cand ≠ ∅ then
8  |  |  SC[S_v] := cand;
9  |  end
10 end
11 return SC;
```

4 Evaluation

4.1 Experimental Setup

All experiments were performed on a machine with a 2.67 GHz Intel Core i5 CPU, 2 GB RAM and running Windows 7 (32 Bit). The simulation environment was written in Java 1.6 using the jMetal 4.0 framework [13].

The settings for each algorithm were the same. Population size was always 100. Workflow length was varied from 10 to 80 with a stepping of 10. Before each simulation run, a process model with local constraints was randomly generated. For each task 20 random service alternatives were generated containing the QoS attributes $price \in]0, 5], responseTime \in]0, 500], location \in \{Brazil, Germany, USA\}$ and $encryption \in \{None, AES-64, AES-128, AES-256\}$. Next, this service set was clustered as discussed in section 3.3. After that, each algorithm performed optimization on this setting 100 times with an allowed maximum of 25,000 evaluations. The algorithms had to minimize a total of three objectives: Price, Response Time and Compliance Distance.

We selected a number of state-of-the-art GAs (IBEA [14], NSGA-II [12] and SPEA2 [15]) which provide overall good results for most optimization problems. Besides, a random approach was utilized to obtain an approximate baseline for our experiments. These algorithms are shipping with jMetal.

4.2 Results

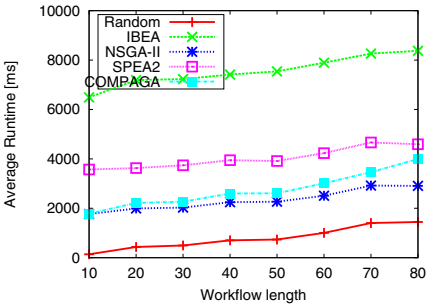
In our experiments we investigated the influence of workflow length on the runtime of each algorithm as well as the average compliance distance, response time and price of determined solutions (see fig. 3).

Runtime: The random approach was naturally the fastest and IBEA the slowest. The runtime of SPEA2 was roughly the median of the random approach and IBEA. Second fastest was NSGA-II. The runtime of COMPAGA was slightly higher than NSGA-II due to the additional repair operation. This difference increased with increasing workflow length due to higher repair efforts.

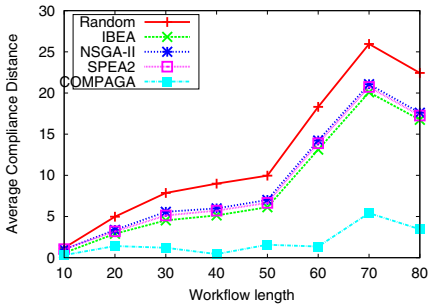
Compliance distance: Average compliance distance generally increased for increasing workflow length except of COMPAGA which delivered significantly better results. IBEA, NSGA-II and SPEA2 yielded similar results with IBEA being slightly better than the other. The random approach performed worst.

Response time: Again, the random approach performed worst. IBEA, NSGA-II and SPEA performed similar with IBEA delivering slightly better solutions. COMPAGA outperformed the remaining algorithms increasingly for increasing workflow length. The reason seams obvious as time constraints are a compliance aspect addressed by the repair operation. Therefore the reduced response time is considered as a side effect of minimizing compliance distance.

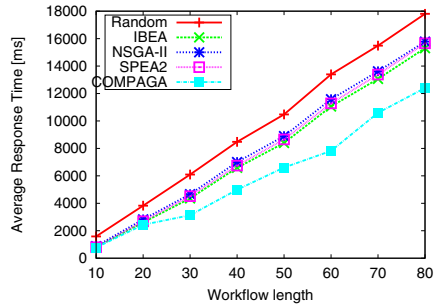
Price: The random approach performed worst while the GAs delivered similar results. COMPAGA performed slightly better for workflow lengths ≥ 40. This seems odd as the price of each service alternative is random. Our explanation



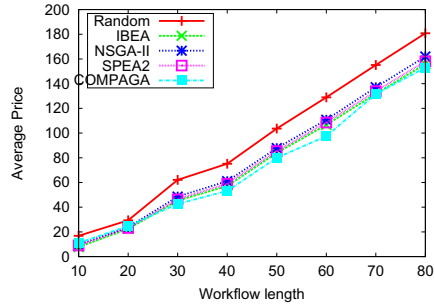
(a) Runtime vs. workflow length



(b) Compliance Distance vs. workflow length



(c) Response time vs. workflow length



(d) Price vs. workflow length

Fig. 3. Evaluation results

is that solutions with lower compliance distance also had a lower price than the other solutions. Therefore we do not attribute this to COMPAGA.

5 Related Work

Approaches for checking process compliance can roughly be divided into three categories: 1) design time approaches for process models, 2) approaches that check compliance during or after system configuration and 3) approaches to verify the compliance of process instances during and after execution. The first group ranges from guiding approaches based upon compliant process patterns (e.g. [16]) to methods for statically checking properties of process models, e.g. based on rule Petri nets (RPNs) (e.g. [17]). Governatori et al. provide a framework which produces a detailed report whether a process model (partly) fulfills compliance rules expressed using the Formal Contract Language (FCL) [18].

Approaches of the second group include formal methods for compliance-aware service composition such as [19]. Another research direction are approaches for verification of service compositions such as [20, 21]. [20] focuses on reachability, liveness and deadlocks. The authors of [21] propose a specification language for compliance properties as well as a verification framework for service compositions

in BPEL. Conceptually our work (and service selection in general) belongs to this group as well. While the aforementioned works represent exact approaches which can be utilized for small and medium-size process models, our approach aims at scenarios with either a) very large process models or b) situations with hard time constrains which require quick reconfiguration of compositions by exchanging services. Although GAs are very common to find near-optimal service compositions in a short period of time (see e.g. [5–7]), it is surprising that to our best knowledge no approach considers compliance issues.

The third group finally consists of approaches which are based upon analyzing data such as logs. This analysis forms the basis for deciding whether a process instance is in conformance with compliance rules or not (see e.g. [22]).

6 Conclusion and Outlook

We discussed which compliance aspects may be effected by service selection with GAs and found that *data*, *location* and *time limits* need to be considered. We then presented an approach to determine compliance violations as well as a method to recover compliance of service compositions based upon the notion of compliance distance. The approach was tested by means of a repair operation with a custom GA called COMPAGA which is based on NSGA-II. Comparisons of COMPAGA with related state-of-the-art GAs on service selection problems showed that COMPAGA clearly outperformed the other GAs in terms of average compliance distance and response time.

Up to this point, COMPAGA differs from NSGA-II only by a repair operator. For the future we want to investigate the potential for improvement by utilizing customized genetic operators that leverage domain-specific knowledge regarding compliance. Since the runtime of COMPAGA increases faster than the runtime of other GAs, increasing the performance of COMPAGA is essential for dealing with workflow lengths $>> 80$ efficiently. Another important field of future research is the question of guaranteeing compliance or to conclude that no compliance is possible for given process models and service alternatives.

Acknowledgments. This work is supported by the German Federal Ministry of Education and Science (BMBF) under grant no. 13N10964 in the project ReSCUeIT. We particularly thank Marcel Heupel and the anonymous reviewers for informative and thorough reviews.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer 40(11), 38–45 (2007)
2. Yu, T., Lin, K.J.: Service selection algorithms for web services with end-to-end qos constraints. In: Proc. IEEE Intl. Conf. on E-Commerce Tech., pp. 129–136 (2004)
3. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (2004)
4. Kharbili, M.E., de Medeiros, A.K.A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: Loos, P., Markus, Nüttgens, et al (eds.) MobIS. LNI, vol. 141, pp. 107–113. GI (2008)

5. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proc. Conference on Genetic and Evolutionary Computation (GECCO), pp. 1069–1075 (2005)
6. Jaeger, M.C., Mühl, G.: Qos-based selection of services: The implementation of a genetic algorithm. In: Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference, pp. 1–12 (2007)
7. Ye, Z., Zhou, X., Bouguettaya, A.: Genetic algorithm based qoS-aware service compositions in cloud computing. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part II. LNCS, vol. 6588, pp. 321–334. Springer, Heidelberg (2011)
8. Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
9. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems. Springer (2012)
10. Curtis, B., Kellner, M.I., Over, J.: Process modeling. Communications of the ACM 35(9), 75–90 (1992)
11. Stohr, E.A., Zhao, J.L.: Workflow automation: Overview and research issues. Information Systems Frontiers 3(3), 281–296 (2001)
12. Deb, K., Pratab, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comp. 6(2), 182–197 (2002)
13. Durillo, J.J., Nebro, A.J.: jMetal: A java framework for multi-objective optimization. Advances in Engineering Software 42(10), 760–771 (2011)
14. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN VIII, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
15. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K.C., et al. (eds.) Evolutionary Methods for Design, Optimisation, and Control, CINME, Barcelona, Spain, pp. 95–100 (2002)
16. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
17. Accorsi, R., Lowis, L., Sato, Y.: Automated certification for compliant cloud-based business processes. BISE 3(3), 145–154 (2011)
18. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: Ardagna, D., Mecella, M., Yang, J. (eds.) RGU 1974. LNBIP, vol. 17, pp. 5–17. Springer, Heidelberg (1974)
19. Bernardi, G., Bugliesi, M., Macedonio, D., Rossi, S.: A theory of adaptable contract-based service composition. In: Proc. 2008 10th Intl. Symp. on Symbolic and Numeric Algorithms for Scientific Computing. SYNASC 2008, pp. 327–334. IEEE Computer Society, Washington, DC (2008)
20. Narayanan, S., McIlraith, S.A.: Simulation, verification and automated composition of web services. In: Proc. 11th Intl. Conf. on World Wide Web (WWW), pp. 77–88 (2002)
21. Yu, J., Manh, T.P., Han, J., Jin, Y., Han, Y., Wang, J.: Pattern based property specification and verification for service composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)
22. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)