STUDIA INFORMATICA
Nr 1-2 (19) Systems and information technology 2015

Jaroslaw Skaruz[1], **Artur Niewiadomski**[1], **Wojciech Penczek**[1,2]

[1] Institute of Computer Science, Siedlce University of Natural Sciences and Humanities, Poland
`{jaroslaw.skaruz},{artur.niewiadomski}@uph.edu.pl`
[2] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
`wpenczek@gmail.com`

Combining SMT and Simulated Annealing into a Hybrid Planning Method*

Abstract. We present a new approach to the concrete planning (CP) - a stage of the Web service composition in the PlanICS framework. A new hybrid algorithm (HSA) based on a combination of Simulated Annealing (SA) with Satisfiability Modulo Theories (SMT) has been designed and implemented. The main idea of our hybrid solution is to use an SMT-based procedure in order to generate an initial individual and then improve it during subsequent iterations of SA. The experimental results show that HSA is superior to the other methods we have applied to the CP problem, including Genetic Algorithm, an SMT-based approach, and our previously developed hybrids.

Keywords: Concrete Planning, Genetic Algorithm, Satisfiability Modulo Theories, Simulated Annealing, Web Service Composition, PlanICS

1 Introduction and Related Work

The main concept of Service-Oriented Architecture (SOA) [3] consists in exploiting autonomous, simple services with well defined interfaces to realize often a sophisticated user objective. The problem of finding such a composition of services is hard and known as the Web Service Composition (WSC) problem [1][3][15].

PlanICS [6] is a framework aimed at WSC dividing the planning into several stages. First, PlanICS makes intensive use of an *ontology*, i.e., a description of some domain of interest as a hierarchy of classes. The classes correspond to the service types, i.e., sets of real-world services, and the types of processed objects. As a result an *abstract plan* composed of service types is produced [10]. The next phases work in the space of *concrete services*. In particular, Offer Collector in cooperation with Service Repository (PlanICS modules) communicate with web services in order to retrieve offers - data used in CP. Overall, CP consists in choosing offers for service types from abstract plan in such a way that all constraints are satisfied and the value of a quality function is maximized. Thus, this can be seen as a variant of constrained optimisation problem. The

* This work has been supported by the National Science Centre under the grant No. 2011/01/B/ST6/01477.

quality function is given as a part of the user query, while the constraints are extracted using the query and dependencies between service types in the abstract plan.

This paper deals with the concrete planning, shown to be NP-hard [11]. Our previous works employ several techniques to solve it: a Genetic Algorithm (GA) [16], numeric optimization methods [13] as well as Satisfiability Modulo Theories (SMT) Solvers [11]. The proposed methods are complementary, but every single one suffers from some disadvantages: The SMT-based solution often needs a long computation time, while the GA approach yields solutions which could be far from optimum and are found with a low probability. Thus, we have combined both the methods and we proposed several versions of a hybrid approach [12].

Over the last few years, the concrete planning problem has been extensively studied in the literature. G. Canfora et al. [4] use GA to obtain a concrete plan. As optimization criteria the authors choose features commonly referred to as Quality of Service (QoS), like the response time of a web service, its cost, availability, and reliability. In [2] the authors use two algorithms to deal with WSC, namely GA and SA. Experiments show better quality of solutions found by SA than GA, but for the price of a higher computation time and lower scalability. The method proposed in [8] is based on the Harmony Search algorithm which selects optimal service compositions faster than a GA-based approach. Hybrid algorithms for WSC are also known in the literature. In [7] a modified version of the Particle Swarm Optimization algorithm is used. Another approach consists in a combination of two evolutionary algorithms, Tabu Search and GA [14].

In what follows we present our solution followed by experimental results and conclusions.

2 Solution

First, we briefly recall the formulation of CP as a constrained optimization problem. We focus on the intuitions only, but the interested reader can see all the formal definitions in [13].

As mentioned above, the input for concrete planning is the result of the offer collecting phase. In particular, it consists of n *offer sets* and a set of *constraints*, where n is the length of the abstract plan under consideration, so each offer set corresponds to some service type. Thus, each offer from a particular set is a tuple of values corresponding to attributes of objects being processed by the given service type. Overall, a solution of CP consists in selecting one offer from each offer set, such that all constraints are satisfied and the value of the quality function Q is maximized.

2.1 Simulated Annealing

SA [9] is an optimisation technique based on an observation of the physical process of annealing in metallurgy, i.e., a technique of heating and slow cooling of a material in order to improve its properties. SA implements the cooling process as a slow decrease in the probability of accepting worse solutions while the algorithm explores the search space. In SA, our "processed material" is a potential solution of a problem, called an individual. The space exploration is realised by applying a neighbourhood operator to the individual, which results in obtaining a new potential solution. In our case, the individual is a sequence of natural values being the offer indices, while the neighbourhood operator changes one value randomly.

Next, if a new individual is better than the current one, i.e., the quality function value of the new one is higher than of the current one, then the new individual becomes the current one.

Otherwise, the worse solution can replace the current one, if the following formula is satisfied:

$$ran < exp(\frac{Q(Inew) - Q(Icur)}{temp}), \qquad (1)$$

where ran is a random value between 0 and 1, $Inew$, $Icur$ stand for the new and the current individual, respectively, Q is a quality function, and $temp \in (0, 1)$ is a temperature value. Worse solutions are more often accepted with a high temperature. The idea of SA is to allow for exploration of the search space by the acceptance of a worse solution at the beginning of the algorithm. Decreasing value of the temperature leads to preferring better solutions.

SA Hybrid Algorithm. It is quite hard to force SA to search for better solutions with satisfied constraints, because employing standard mechanisms like penalty functions known from GA is problematic. Thus, our algorithm applies an SMT-based procedure (similar to the one described in [12]) to generate an initial individual which satisfies all constraints, and the next steps of our hybrid algorithm are almost of a standard SA. The main difference is that new individuals may be accepted provided that all constraints are still satisfied. Since an initial individual is already a solution (usually of a low quality), the HSA algorithm always returns a result, and the objective is to increase the value of the quality function. The algorithm uses the following parameter values: a number of iterations of the main (G) and internal loop (IL), an initial value of the temperature ($temp$), and the temperature decreasing factor ($decFactor$). The pseudocode of HSA is presented as Algorithm 1.

Experiments. We have evaluated the efficiency of HSA using the same six benchmarks as in the papers [13, 12]. All the instances represent plans of length 15. Each offer set of Instance 1, 3, and 5 contains 256 offers, which makes the number of the potential solutions equal to $256^{15} = 2^{120}$. In the case of Instance 2, 4, and 6, each offer set consists of 512 offers, which results in the search space size as large as $512^{15} = 2^{135}$.

The values of parameters of HSA are as follows: $temp = 1.0$, $G = 500$, $IL = 40$, $decFactor = 0.98$. Every experiment has been repeated 50 times on a standard PC with 2.8GHz CPU and Z3 [5] version 4.3 as the SMT-solving engine.

In Table 1 we compare the performance of HSA with our previous results. The first column contains the instance number, while the next columns present computation times and the quality function values of the solutions found with our different methods: HSA, IPH (in two variants - with 1 and 500 individuals generated by SMT), and

```
begin
    generate an initial individual $Icur$ by the SMT-based procedure;
    calculate the quality function for the initial individual $Q(Icur)$;
    for $i := 1$ to $G$ $(generations)$ do
        for $j := 1$ to $IL$ (internal loop) do
            generate a new individual $Inew$ from neighbourhood;
            if $Inew$ satisfies all constraints then
                calculate the quality function for the new individual $Q(Inew)$;
                if $Q(Inew) > Q(Icur)$ OR $ran < exp(\frac{Q(Inew)-Q(Icur)}{temp})$ then
                    $Icur = Inew$;
                end
            end
        end
        $temp = temp * decFactor$;
    end
    return $Icur$;
end
```

Algorithm 1: Pseudocode of HSA for concrete planning.

non-hybrid ones: SMT and GA. The HSA algorithm is the fastest one and finds solutions of reasonable quality. It has to be mentioned that all the presented methods return solutions with 100% probability, except for GA, where the probability is much lower: from 7% to 12%. The results are also summarised in Fig. 1 which presents values of quality function divided by computation time.

Table 1. Experimental results of the IPH and HSA algorithms.

	HSA		**IPH$_1$**		**IPH$_{500}$**		**SMT**		**GA**	
Instance	t[s]	Q	t[s]	Q	t[s]	Q	t[s]	Q	t[s]	Q
1	4.0	1420	5.6	1229.5	13.9	1317.9	266.0	1443.0	5.0	1218.5
2	5.1	1402	6.4	1248.8	20.1	1382.4	388.0	1467.0	5.6	1319.9
3	5.4	2153	6.4	1706.8	14.6	2090.6	500.0	2266.0	6.0	2085.4
4	6.5	2194	7.5	1788.0	20.9	2280.3	500.0	2409.0	6.6	2001.9
5	4.7	354	6.4	329.1	27.8	634.5	500.0	781.0	5.1	436.0
6	6.3	541	8.8	458.2	55.2	524.7	500.0	755.0	5.9	537.8

3 Conclusions

We presented a new approach to concrete planning. Because a standard method based on an application of a penalty function used in constrained optimization problems is

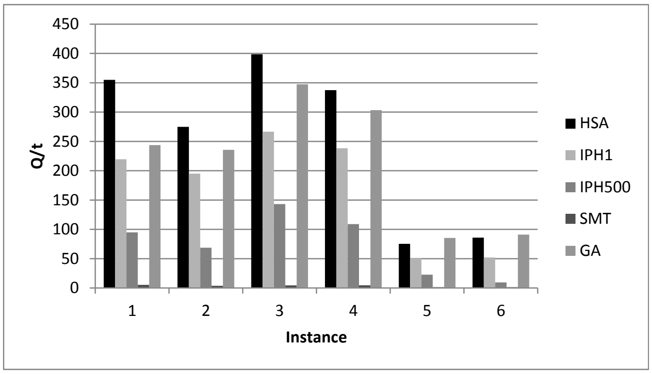

Figure 1. Comparison of different planning methods efficiency.

not satisfactory in SA, we employed SMT for generating an initial individual. Such an individual is already a solution because it satisfies all the constraints. During the next steps of our algorithm the individual is improved in order to find a concrete plan with a greater value of the quality function. The experiments confirm that HSA is efficient, since the results are better than those obtained by our other hybrid algorithms.

References

1. S. Ambroszkiewicz. Entish: A language for describing data processing in open distributed systems. *Fundam. Inform.*, 60(1-4):41–66, 2003.
2. L. Arockiam and N. Sasikaladevi. Simulated annealing versus genetic based service selection algorithms. *International Journal of u- and e- Service, Science and Technology*, 5:35–49, 2012.
3. M. Bell. *Introduction to Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture*. John Wiley & Sons, 2008.
4. G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 1069–1075, 2005.
5. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. of TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer-Verlag, 2008.
6. D. Doliwa et al. PlanICS - a web service compositon toolset. *Fundam. Inform.*, 112(1):47–71, 2011.
7. C. Hu, X. Chen, and X. Liang. Dynamic services selection algorithm in web services composition supporting cross-enterprises collaboration. *Journal of Central South University of Technology*, 16:269–274, 2009.
8. N. Jafarpour and M. Khayyambashi. Qos-aware selection of web service compositions using harmony search algorithm. *Journal of Digital Information Management*, 8:160–166, 2010.
9. S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. *Sci.*, 220:671–680, 1983.

10. A. Niewiadomski and W. Penczek. Towards SMT-based Abstract Planning in PlanICS Ontology. In *Proc. of KEOD 2013 – International Conference on Knowledge Engineering and Ontology Development*, pages 123–131, September 2013.
11. A. Niewiadomski, W. Penczek, and J. Skaruz. SMT vs genetic algorithms: Concrete planning in PlanICS framework. In *CS&P*, pages 309–321, 2013.
12. A. Niewiadomski, W. Penczek, and J. Skaruz. A hybrid approach to web service composition problem in the PlanICS framework. In Irfan Awan, Muhammad Younas, Xavier Franch, and Carme Quer, editors, *Mobile Web Information Systems*, volume 8640 of *Lecture Notes in Computer Science*, pages 17–28. Springer International Publishing, 2014.
13. A. Niewiadomski, W. Penczek, J. Skaruz, M. Szreter, and M. Jarocki. SMT versus Genetic and OpenOpt Algorithms: Concrete Planning in the PlanICS Framework. *Fundamenta Informaticae*, 135(4):451–466, 2014.
14. J. A. Parejo, P. Fernandez, and A. R. Cortes. Qos-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingenieria del Software y Bases de Datos*, 2(1):55–66, 2008.
15. J. Rao and X. Su. A survey of automated web service composition methods. In *Proc. of SWSWPC'04*, volume 3387 of *LNCS*, pages 43–54. Springer, 2005.
16. J. Skaruz, A. Niewiadomski, and W. Penczek. Automated abstract planning with use of genetic algorithms. In *GECCO (Companion)*, pages 129–130, 2013.