A Soft Constraint-Based Approach to QoS-Aware Service Selection [⋆]

Mohamed Anis Zemni[1], Salima Benbernou[1], Manuel Carro[2]

[1] LIPADE, Université Paris Descartes, France
[2] Facultad de Informática, Universidad Politécnica de Madrid, Spain
mohamedaniszemni@gmail.com, salima.benbenrou@paridescartes.fr, mcarro@fi.upm.es

Abstract. Service-based systems should be able to dynamically seek replacements for faulty or underperforming services, thus performing self-healing. It may however be the case that available services do not match all requirements, leading the system to grind to a halt. In similar situations it would be better to choose alternative candidates which, while not fulfilling all the constraints, allow the system to proceed. *Soft constraints*, instead of the traditional *crisp* constraints, can help naturally model and solve replacement problems of this sort. In this work we apply soft constraints to model SLAs and to decide how to rebuild compositions which may not satisfy all the requirements, in order not to completely stop running systems.
Keywords: Service Level Agreement, Soft Constraints.

1 Introduction

A (web) service can be defined as a remotely accessible software implementation of a resource, identified by a URL. A set of protocols and standards, such as WSDL, facilitate invocation and information exchange in heterogeneous environments. Software services expose not only functional characteristics, but also non-functional attributes describing their Quality of Service (QoS) such as availability, reputation, etc. Due to the increasing agreement on the implementation and management of the functional aspects of services, interest is shifting towards non-functional attributes describing the QoS. Establishing QoS contracts, described in the Service Level Agreement (SLA), that can be monitored at runtime, is therefore of paramount importance. Various techniques [1] to select services fulfilling functional and non-functional requirements have been explored, some of them based on expressing these requirements as a constraint solving problem [2, 3] (CSP). Traditional CSPs can either be fully solved (when all requirements are satisfied) or not solved at all (some requirements cannot be satisfied). In real-life cases, however, over-constraining is common (e.g., because available services offer a quality below that required by the composition), and problems are likely not to have a classical, crisp solution. Solving techniques for *soft* CSPs (SCSP) [4–6] can generate solutions for overconstrained problems by allowing some constraints to remain unsatisfied.

[⋆] The research leading to these results has received funds from the European Community's Seventh Framework Programme FP7/2007-20013 under grant agreement 215483 (S-CUBE). Manuel Carro was also partially supported by Spanish MEC project 2008-05624/TIN *DOVES* and CM project P2009/TIC/1465 (*PROMETIDOS*).

A C-semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ s.t.

- A is a set and $\mathbf{0} \in A$, $\mathbf{1} \in A$.
- $\sum$ (the *additive* operation)[a] is defined on subsets of A as follows:
  - $+$ is commutative ($a + b = b + a$), associative ($a + (b + c) = (a + b) + c$), with unit element $\mathbf{0}$ ($a + \mathbf{0} = a$) and absorbing element $\mathbf{1}$ ($a + \mathbf{1} = \mathbf{1}$).
  - $\sum \emptyset = \mathbf{0}$ and for all $a \in A, \sum\{a\} = a$.
  - Given any set of indices S, $\sum_{i \in S}(\bigcup A_i) = \sum(\{\sum_{i \in S} A_i\})$ (flattening).
- $\times$ (the *multiplicative* operation) is associative, commutative, $a \times \mathbf{1} = a$ and $a \times \mathbf{0} = \mathbf{0}$.
- $\times$ distributes over $+$, i.e., $a \times (b + c) = (a \times b) + (a \times c)$.

[a] Written as infix $+$ when applied to a two-element set.

Fig. 1. Definition of a C-Semiring for Soft Constraints.

Our framework takes into consideration the penalties agreed upon on the SLA by building a new (Soft) Service Level Agreement (SSLA) based on preferences where strict customer requirements are replaced by soft requirements allowing a suitable composition. This agreement has to include penalty terms to be applied while the contract terms are violated.

2 Soft Constraints in a Nutshell

A CSP defines a set of variables whose ranges we assume a finite domain (FD)[3] and a set of constraints which restrict the values these variables can take. A solution for a CSP is an assignment of a value to *every* variable s.t. all the constraints are simultaneously satisfied. Soft constraints [5, 6] generalize classical CSPs by adding a preference level to every tuple in the domain of the constraint variables. This level can be used to obtain a suitable solution which may not fulfill all constraints, which optimizes some metrics, and which in our case will be naturally applied to the requirements of the users.

The basic operations on soft constraints (building a constraint conjunctions and projecting on variables) need to handle preferences in a homogeneous way. This requires the underlying mathematical structure of classical CSPs to change from a cylindrical algebra to a semiring algebra, enriched with additional properties and termed a *C-semiring* (Figure 1). In it, A provides the levels of preference of the solutions and it can be proved that it is a lattice with partial order $a \preceq b$ iff $a + b = b$, minimum $\mathbf{0}$, and maximum $\mathbf{1}$. When solutions are combined or compared, preferences are accordingly managed using the operations $\times$ and $+$. Note that the theory makes no assumptions as to what the preferences mean, or how they are actually handled: $\times$ and $+$ are placeholders for concrete definitions which can give rise to different constraint systems, such as fuzzy constraints, traditional constraints, etc.

Figure 2 summarizes some basic definitions regarding soft constraints. A constraint takes a tuple of variables and assigns it a tuple of concrete values in the domain of the

[3] CSPs can be defined on infinite domains, but assume a FD here because it can accommodate many real-life problems, as witnessed by the relevance of FD in industrial applications, and because soft constraint theory requires finiteness.

Definition 1 (Constraint). *Given a c-semiring $\langle A, +, \times, 0, 1\rangle$, a set of variables V, and a set of domains D, one for every variable in V, a constraint is the pair $\langle def, con\rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow A$.*

Definition 2 (Soft Constraint Satisfaction Problem SCSP). *A SCSP is a pair $\langle C, con\rangle$ where $con \subseteq V$ and C is a set of constraints. C may contain variables which are not in con, i.e., they are not interesting for the final result. In this case the constraints in C have to be projected onto the variables in con.*

Definition 3 (Constraint combination). *Two constraints $c_1\langle def_1, con_1\rangle$ and $c_2 = \langle def_2, con_2\rangle$ can be combined in $c_1 \otimes c_2 = \langle def, con\rangle$ by taking all the variables in the original constraints ($con = con_1 \bigcup con_2$) and assigning to every tuple in the new constraint a preference value which comes from combining the values in the original constraints: $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$, with $t \downarrow^{X}_{Y}$ denoting the projection of tuple t, which is defined on the set of variables X, over the set of variables $Y \subseteq X$.*

Definition 4 (Projection). *Given a soft constraint $c = \langle def, con\rangle$ and a set of variables $I \subseteq V$, the projection of c over I, denoted by $c \Downarrow_I$ is the constraint $\langle def', con'\rangle$ where $con' = con \bigcap I$ and $def'(t') = \sum_{\{t|t\downarrow^{con}_{con\cap I}=t'\}} def(t)$.*

Definition 5 (Solution). *A solution of a SCSP problem $\langle C, con\rangle$ is the constraint $(\otimes C) \Downarrow_{con}$, i.e., the combination (conjunction) of all the constraints in C projected over all the variables con of interest.*

Fig. 2. Definitions for Soft Constraints.

variables, plus a preference value (belonging to the set A). Constraints can be combined into other constraints (with $\otimes$, similar to conjunction) and projected ($\Downarrow^{X}_{Y}$) onto a tuple of variables. The preference value of every tuple in a constraint conjunction is worked out by applying $\times$ to the preference values of the tuples in the individual constraints. Projections eliminate "columns" from tuples and retain only the non-removed tuple components. Repeated rows may then appear, but only one is retained, and its preference is calculated applying $+$ to the preferences of the repeated tuples. Since a solution is a projection on some selected set of variables, the preferences of solutions are naturally calculated using the projection operation. Usually the tuple with the highest preference value is selected as the "optimal" solution.

3 Soft Service Level Agreement and SCSPs

A Service Level Agreement (SLA) [7] is a contract between provider(s) and client(s) specifying the guarantees of a service, the expected quality level, and the penalties to be applied in case of unfulfillment of duties, and it is therefore an important quality management artifact. The SLA can be used to identify the responsible of a malfunction and to decide which action (if any) has to be taken. An SLA should, therefore, be realistic, achievable, and maintainable.

An SLA has a rich structure from which we underline the properties of the services, including those measurable aimed at expressing guarantees. This part provides a set $\varUpsilon$ of variables υ_i (whose meaning is explained in the service description) and their domains $\delta_i \in \varDelta$, which can be established by the metric attribute. A Soft SLA (SSLA) is similar to a SLA but with the addition of a set of user preferences and of penalties

Definition 6 (Preference). *The set $Pr = \{\langle \delta_i, \upsilon_i, a_i \rangle | \delta_i \in \Delta, \upsilon_i \in \Upsilon, a_i \in A\}$ where δ_i is the sub-domain that the i-th preference belongs to, υ_i is the variable defining the preferences, and a_i is semiring value, representing the preferences in an SSLA.*

Definition 7 (Penalty). *The set $Pn = \{pn_i \mid \exists pr_i \; s.t. \; \upsilon_i \notin \delta_i\}$ represents the penalties.*

Definition 8 (SSLA document). *A SSLA document is a tuple $\zeta = \langle \Upsilon, \Delta, A, Pr, Pn, T \rangle$ where Υ is a set of variables υ_i, Δ is a set of variable domains δ_i (one for each variable), Pr is a set of preferences Pr_i, Pn is a set of penalties Pn_i to apply when the preferences are not satisfied and T is a set of pairs $\langle pr_i, pn_i \rangle$ which associates preferences with the penalties to apply in case of violation.*

Fig. 3. Definitions related to a soft SLA.

associated to contract breaking (respectively, Pr and Pn). The preferences are used to make a composition in the presence of unsatisfied requirements and the penalties are used to refine found solutions and to protect each party from the violation of the contract terms. These notions are depicted in Figure 3. The i-th penalty $pn_i \in Pn$ is applied when the i-th preference $pr_i \in Pr$ is not satisfied.

3.1 Extending SCSP Using Penalties

We will adapt the SCSP framework to handle explicitly penalties for service selection and to build a Soft Service Level Agreement including preferences and penalties. In this framework, service selection has three phases:

1. Model the characteristics for the selection using soft constraints.
2. Assuming a pre-selection is made using functional requirements, rank candidate services using non-functional requirements and the constraint preferences.
3. We assign penalties to unmet user preferences, and these penalties are used to rank solutions having the same constraint preferences.

Figure 4 shows the definitions for this extended SCSP framework. We extend the application of semiring operations to penalties. Variables are assumed to take values over subdomains which discretize a continuous domain, and which for brevity we represent using identifiers in $D_{\{\}}$. The constraint preference function *def* is also adapted in order to apply it both to preferences and to penalties. The projection operation is kept as in the SCSP framework.

3.2 An Example

A delivery service has an order-tracking web service. Companies wishing to hire this service want to have in the contract non-functional criteria such as availability, reputation, response time and cost.

Phase 1 Let $CS = \langle S_p, D_{\{\}}, V \rangle$ be a constraint system and $P = \langle C, con \rangle$ be the problem to be solved, where $V = con = \{\underline{\text{A}}\text{vailability}, \underline{\text{R}}\text{eputation}, \text{response } \underline{\text{T}}\text{ime}, \text{co}\underline{\text{S}}\text{t}\}$, $D_{\{\}} = \{\{a_1, a_2\}, \{r_1, r_2\}, \{t_1, t_2, t_3\}, \{s_1, s_2\}\}$, $S_p = \langle [0,1], Pn, \max, \min, 0, 1 \rangle$, $C = \{c_1, c_2, c_3, c_4\}$. For simplicity, variables and their domains have been written in the same order.

Definition 9 (CP-semiring). *A CP-semiring is a tuple $S = \langle A, Pn, +, \times, 0, 1\rangle$, extending a C-semiring. A and Pn are two sets with lattice structure stating preference values for solutions and penalties. Operations $\times$ and $+$ are applied when constraints are combined or projected.*

Definition 10 (Constraint System). *A constraint system is a tuple $CS = \langle S, D_{\{\}}, V\rangle$, where S is c-semiring, $D_{\{\}}$ represents the set of identifiers of subdomains, and V is the ordered set of variables.*

Definition 11 (Constraint). *Given a constraint system $CS = \langle S_p, D_{\{\}}, V\rangle$ and a problem $P = \langle C, con\rangle$, a constraint is the tuple $c = \langle def_c, type\rangle$, where $type$ represents the type of constraint and def_c is the definition function of the constraint, which returns the tuple*

$$def : D_{\{\}}^{|con|} \rightarrow \langle p_r, p_n\rangle,$$

Definition 12 (Soft Constraint Satisfaction Problem SCSP). *Given a constraint system $CS = \langle S, D_{\{\}}, V\rangle$, an SCSP over CS is a pair $P = \langle C, con\rangle$, where con, called set of variables of interest for C, is a subset of V and C is a finite set of constraints, which may contain some constraints defined on variables not in con.*

Fig. 4. CP-Semiring.

A set of penalties, ranked from the most to then less important one, has been set: $pn_i \preceq pn_j$ if $i \leq j$. The above shown values of the variable domains comes from a discretization such as $availability \in \{[0, 0.5[, [0.5, 1]\}$, $reputation \in \{[0, 0.6[, [0.6, 1]\}$, $response\ time \in \{[20, \infty[, [5, 20[, [0, 5[\}$, $cost \in \{[1000, 1500[, [1500, 3000]\}$.

Let us consider the following constraints: $c_1 = \langle def_{c1}, \{\text{availability, reputation}\}\rangle$, $c_2 = \langle def_{c2}, \{\text{response time}\}\rangle$, $c_3 = \langle def_{c3}, \{\text{availability, reputation, cost}\}\rangle$, $c_4 = \langle def_{c4}, \{\text{reputation, response time}\}\rangle$, where the preference values and corresponding penalties are in Table 1. For example, for the tuple $\langle a_2, r_1\rangle$, attributes "availability" and "reputation" are respectively assigned subdomains $[0.5, 1]$ and $[0, 0.6[$. The function $def_{c1}(\langle a_2, r_1\rangle) = \langle 0.5, pn_3\rangle$ shows that these attribute values have a preference 0.5 and company is ready to sign away this preference for a penalty pn_3.

Phase 2 Given the model, we define constraint combination to keep the minimum value of preferences (resp. for the penalties). For example $def_{c1}(\langle a_2, r_1\rangle) \otimes def_{c2}(\langle t_3\rangle) = \min(\langle 0.5, pn_3\rangle, \langle 0.25, pn_6\rangle) = \langle 0.25, pn_3\rangle$ and so on with all the tuples to obtain $c_{1,2}$. Next, we would combine $c_{1,2}$ and c_3 to get $c_{1,2,3} = c_{1,2} \otimes c_3$ and so on, until all constraints have been combined. Table 2 shows the results of combining all the constraints.

$\langle A, R\rangle$	def_{c1}	$\langle T\rangle$	def_{c2}	$\langle A, R, S\rangle$	def_{c3}	$\langle R, T\rangle$	def_{c4}
$\langle a_1, r_1\rangle$	$\langle 0, -\rangle$	$\langle t_1\rangle$	$\langle 0.25, pn_6\rangle$	$\langle a_1, r_1, s_1\rangle$	$\langle 0.25, pn_8\rangle$	$\langle r_1, t_1\rangle$	$\langle 0.5, pn_6\rangle$
$\langle a_1, r_2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle t_2\rangle$	$\langle 0.5, pn_5\rangle$	$\langle a_1, r_1, s_2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle r_1, t_2\rangle$	$\langle 0.5, pn_5\rangle$
$\langle a_2, r_1\rangle$	$\langle 0.5, pn_3\rangle$	$\langle t_3\rangle$	$\langle 1, pn_7\rangle$	$\langle a_1, r_2, s_1\rangle$	$\langle 0.5, pn_1\rangle$	$\langle r_1, t_3\rangle$	$\langle 0, -\rangle$
$\langle a_2, r_2\rangle$	$\langle 0.75, pn_3\rangle$			$\langle a_1, r_2, s_2\rangle$	$\langle 0.25, pn_3\rangle$	$\langle r_2, t_1\rangle$	$\langle 0.75, pn_2\rangle$
				$\langle a_2, r_1, s_1\rangle$	$\langle 0.75, pn_9\rangle$	$\langle r_2, t_2\rangle$	$\langle 0.75, pn_4\rangle$
				$\langle a_2, r_1, s_2\rangle$	$\langle 0.5, pn_8\rangle$	$\langle r_2, t_3\rangle$	$\langle 1, pn_2\rangle$
				$\langle a_2, r_2, s_1\rangle$	$\langle 0.75, pn_2\rangle$		
				$\langle a_2, r_2, s_2\rangle$	$\langle 0.25, pn_1\rangle$		

Table 1. Constraint definitions.

$\langle A, R, T, S\rangle$	$\langle pr, pn\rangle$	$\langle A, R, T, S\rangle$	$\langle pr, pn\rangle$	$\langle A, R, T, S\rangle$	$\langle pr, pn\rangle$	$\langle A, R, T, S\rangle$	$\langle pr, pn\rangle$
$\langle 2,2,3,1\rangle$	$\langle 0.75, pn_2\rangle$	$\langle 2,2,1,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 1,2,1,1\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 1,1,3,2\rangle$	$\langle 0.0, -\rangle$
$\langle 2,2,2,1\rangle$	$\langle 0.50, pn_2\rangle$	$\langle 1,2,3,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 2,2,1,1\rangle$	$\langle 0.25, pn_2\rangle$	$\langle 1,1,3,1\rangle$	$\langle 0.0, -\rangle$
$\langle 2,1,2,2\rangle$	$\langle 0.50, pn_3\rangle$	$\langle 1,2,3,1\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 2,1,1,2\rangle$	$\langle 0.25, pn_3\rangle$	$\langle 1,1,2,2\rangle$	$\langle 0.0, -\rangle$
$\langle 2,1,2,1\rangle$	$\langle 0.50, pn_3\rangle$	$\langle 1,2,2,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 2,1,1,1\rangle$	$\langle 0.25, pn_3\rangle$	$\langle 1,1,2,1\rangle$	$\langle 0.0, -\rangle$
$\langle 2,2,2,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 1,2,2,1\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 2,1,3,2\rangle$	$\langle 0.0, -\rangle$	$\langle 1,1,1,2\rangle$	$\langle 0.0, -\rangle$
$\langle 2,2,3,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 1,2,1,2\rangle$	$\langle 0.25, pn_1\rangle$	$\langle 2,1,3,1\rangle$	$\langle 0.0, -\rangle$	$\langle 1,1,1,1\rangle$	$\langle 0.0, -\rangle$

Table 2. Ordered constraint combinations with preferences and penalties.

Phase 3 The set of solutions is ranked by preferences and then by penalties (already in Table 2). The solution with highest rank is chosen first. If it turns out not to be feasible, the associated penalty is applied and the next solution is chosen, and so on.

3.3 Mapping SSLA onto SCSP Solvers

Given the our design of an SSLA, mapping it into a SCSP is very easy: variables v_i in the SSLA are mapped onto the corresponding v_i in the SCSP; SSLA domains δ_i are discretized and every discrete identifier is a domain for a SCSP variable; and preferences and penalties (both lattices) are handled together by the def function, so they can be mapped to the A set in a C-semiring with an adequate definition of the def function.

4 Conclusion

We have presented a soft constraint-based framework to seamlessly express QoS properties reflecting both customer preferences and penalties applied to unfitting situations. The application of soft constraints makes it possible to work around overconstrained problems and offer a feasible solution. Our approach makes easier this activity thanks to ranked choices. Introducing the concept of penalty in the Classical SCSP can also be useful during the finding and matching process. We plan to extend this framework to also deal with behavioral penalties.

References

1. Carlos Müller, Antonio Ruiz-Cortés, and Manuel Resinas. An Initial Approach to Explaining SLA Inconsistencies. In Athman Bouguettaya, Ingolf Krueger, and Tiziana Margaria, editors, *Service-Oriented Computing (ICSOC 2008)*, volume 5364 of *LNCS*, pages 394–406, 2008.
2. Ugo Montanari. Networks of Constraints: Fundamental Properties and Application to Picture Processing. *Information Sciences 7*, pages 95 – 132, 1974.
3. Rina Dechter. *Constraint Processing*. Morgan Kaufman, 2003.
4. S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*. Springer, 2004.
5. Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.
6. S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. *In Proc. IJCAI95*, 1995.
7. Philip Bianco, Grace A.Lewis, and Paulo Merson. Service Level Agreements in Service-Oriented Architecture Environment. Technical Report CMU/SEI-2008-TN-021, Carnegie Mellon, September 2008.