Resolving Underconstrained and Overconstrained Systems of Conjunctive Constraints for Service Requests

Muhammed J. Al-Muhammed* and David W. Embley*

Department of Computer Science
Brigham Young University, Provo, Utah 84602, U.S.A.

Abstract. Given a service request such as scheduling an appointment or purchasing a product, it is possible that the invocation of the service results in too many solutions that all satisfy the constraints of the request or in no solution that satisfies all the constraints. When the invocation results in too many solutions or no solution, a resolution process becomes necessary for agreeing on one of the solutions or finding some agreeable resolution. We address this problem by imposing an ordering over all solutions and over all near solutions. This ordering provides a way to select the best-m with dominated solutions or dominated near solutions eliminated. Further, we provide an expectation-based resolution process that can take the initiative and either elicit additional constraints or suggest which constraints should be relaxed. Experiments with our prototype implementation show that this resolution process correlates substantially with human behavior and thus can be effective in helping users reach an acceptable resolution for their service requests.

Keywords: Service requests, underconstrained systems of constraints, overconstrained systems of constraints, ordered solutions and near solutions, dominance, expectation-based resolution.

1 Introduction

We described in a previous paper [AMEL05] a system that allows users to specify service requests and invoke services. This approach is strongly based on conceptual modeling and supports a particular type of service whose invocation involves establishing an agreed-upon relationship in the conceptual model. Examples of these types of services include scheduling appointments, setting up meetings, selling and purchasing products, making travel arrangements, and many more. It is possible that the invocation of service requests for any of these services results in too many satisfying solutions or in no solution at all although there may be near solutions.

In our approach users can specify services such as the following request for scheduling an appointment with a dermatologist.

* Supported in part by the National Science Foundation under grants 0083127 and 0414644.

> I want to see a dermatologist on the 20th, 1:00 PM or after. The dermatologist should be within 5 miles from my home and must accept my IHC insurance.

Our approach uses conceptual-model-based information extraction to map service requests to a domain ontology. This mapping transforms the service request into a formal representation, which consists of concepts along with relationships among these concepts and constraints over the values of these concepts in a domain ontology. Figure 1 shows the formal representation of the appointment request as a conjunctive predicate calculus statement—we added some comments prefixed with "//" to provide more readability and to correlate the request with the predicate calculus statement. To resolve the appointment request, the system tries to instantiate each variable in the formal representation with values such that all the constraints are satisfied. The values come from a databases associated with the domain ontology, or are extracted from the service request, or are obtained interactively from users.[1]

$$
\begin{array}{l}
//I\ want\ to\ see\ a\ dermatologist \\
Appointment(x_0)\, is\, with\, Dermatologist(x_1) \wedge Appointment(x_0)\, is\, for\, Person(x_2) \\
//on\ the\ 20th \\
\wedge\, Appointment(x_0)\, is\, on\, Date(\text{“}the\ 20th\text{”}) \\
//1{:}00\ PM\ or\ after \\
\wedge\, Appointment(x_0)\, is\, at\, Time(x_3) \wedge TimeAtOrAfter(x_3,\ \text{“}1{:}00\text{”}) \\
//within\ 5\ miles\ from\ my\ home \\
\wedge\, Dermatologist(x_1)\, is\, at\, Address(x_4) \wedge Person(x_2)\, is\, at\, Address(x_5) \\
\wedge\, LessThanOrEqual(DistanceBetween(x_4, x_5),\ \text{“}5\text{”}) \\
//accept\ my\ IHC\ insurance \\
\wedge\, Dermatologist(x_1)\, accepts\, Insurance(\text{“}IHC\text{”})
\end{array}
$$

Fig. 1. The predicate calculus statement for the appointment request.

A *solution* for a request is an instantiation for all the variables that satisfies all the constraints. A *near solution* is an instantiation for all the variables that satisfies a proper subset (maybe empty) of the constraints and, in a way to be made precise later, comes close to satisfying the constraints not (yet) satisfied. Ideally, our system would find just one solution or would find a handful of solutions from which a user could select a desired one. More typically, however, our system may return no solution or too many solutions. When our system returns no solution, the request is *overconstrained*, and when it returns too many solutions, the request is *underconstrained*.

A resolution for overconstrained requests is to offer the best-m near solutions. Figure 2 shows three near solutions for our appointment request. Both s_1 and s_2 violate the date and distance constraints at different degrees in the sense

[1] The details of producing formal representations and instantiating them are not the focus of this paper and can be found elsewhere [AMEL05].

	Date	Time	Distance
s_1	the 21th	1:00 PM	6 miles
s_2	the 22th	1:30 PM	8 miles
s_3	the 20th	2:20 PM	20 miles

Fig. 2. Near solutions for the appointment request.

	Make	Price	Year	Mileage
s_1	Dodge	$13,999	2005	15,775 miles
s_2	Dodge	$13,999	2004	30,038 miles

Fig. 3. Solutions for the car purchase request.

that s_1 is closer to the 20th and violates the distance constraint less than s_2. Consequently, it is reasonable to impose a greater penalty on s_2 than on s_1. Further, the penalty provides a way to recognize dominated near solutions. Near solution s_1 dominates near solution s_2 because s_1 has less of a penalty for each violated constraint. Penalties provide a way to offer the best-m near solutions by ordering the near solutions based on their penalties and discarding the dominated ones. Additionally, suggesting constraints for users to relax provides another way to offer the best-m near solutions. For instance, if prior appointment requests reveal that users are more likely to impose constraints on date and time than on distance, it makes sense to suggest that users relax constraints on distance. Thus, for example, the resolution process can suggest the relaxation of the constraint on distance and possibly offer s_3 as the best near solution in Figure 2.

A resolution for underconstrained requests is to offer the best-m solutions. Consider, for example, the following request for a car purchase.

> I want to buy a Dodge, a 2002 or newer. The mileage should be less than 80,000, and the price should not be more than $15,000.

For this request, www.cars.com offered 168 solutions when probed in November 2005, two of which are in Figure 3. Presenting all the solutions or m arbitrarily chosen ones to users is not likely to be very helpful. A way to reduce the number of solutions and offer the best-m solutions is to elicit additional constraints. If prior car purchase requests reveal that users often impose constraints on the car model, for example, it makes sense that a resolution process elicits a constraint on the model of the car. In addition, some solutions satisfy constraints better than others. As Figure 3 shows, s_1 better satisfies the year constraint than s_2 because the car in s_1 is newer. Therefore, we can grant s_1 a reward for better satisfying the request. Further, the reward can provide a way to recognize dominated solutions. As Figure 3 shows, the solution s_2 is dominated by s_1 because the car in s_1 is newer and has less mileage although both have the same price. Rewards provide a way to offer the best-m solutions by ordering the solutions in a decreasing order based on their rewards and discarding the dominated ones.

This paper offers ways to handle underconstrained and overconstrained service requests. First, the paper offers an expectation-based process for eliciting additional constraints for underconstrained requests and for suggesting some

constraints for users to relax for overconstrained requests. Second, the paper offers an ordering over solutions and an ordering over near solutions, and a selection mechanism based on Pareto optimality [Par97,Fel80], developed in the late 1800's, to choose the best-m, with dominated solutions or dominated near solutions discarded.

We present these contributions as follows. Section 2 discusses an extension to constraints that allows for ordering solutions based on the degree of satisfiability and for ordering near solutions based on how close they are to satisfying the constraints. For underconstrained requests, Section 3 introduces expectation declarations as domain knowledge and proposes an expectation-based process to select concepts for which to elicit constraints. In addition, we define an ordering of solutions based on the extension to constraint satisfaction introduced in Section 2 and use it along with Pareto optimality to select the best-m solutions. For overconstrained requests, Section 4 shows how to define an ordering over near solutions and use it along with Pareto optimality to select the best-m near solutions. It also introduces an expectation-based process to suggest constraints for users to relax. We evaluate our proposed techniques in Section 5, and give concluding remarks and directions for future work in Section 6.

2 Constraints

A *constraint* is an n-place predicate, which for a tuple t of n values evaluates to either *true* or *false* depending on whether t satisfies or violates the constraint. This *true-false* binary view of a constraint allows us to only differentiate tuples based on whether they satisfy or violate a constraint. Researchers have extended this view to differentiate between tuples that violate a constraint by assigning to these tuples increasing positive real numbers that represent different degrees of violation [LHL97,Arn02]. Although this extension allows for distinguishing between tuples that violate a constraint, it does not allow for distinguishing between tuples that satisfy a constraint because this extension lacks the notion of degree of satisfiability. A constraint evaluates to zero for all tuples that satisfy that constraint, which means all the tuples necessarily have the same degree of satisfiability. We, therefore, further extend the binary view to not only consider degree of violation, but also to consider degree of satisfiability by granting tuples increasing rewards based on how well they satisfy a constraint.

Definition 1. *Let C be an n-place constraint and let D_i be the domain of the i^{th} place of C, $1 \leq i \leq n$. A constraint is a function $C : D_1 \times ... \times D_n \longrightarrow \mathcal{R}$ that maps a tuple $t = \langle v_1, ..., v_n \rangle \in D_1 \times ... \times D_n$ to a real number in $\mathcal{R}$. An evaluation of the constraint C on a tuple t is defined as $C(t) = \alpha$, where $\alpha \in \mathcal{R}^+ \cup \{0\}$, which is a positive real number $\mathcal{R}^+$ or zero, is the value of the evaluation if t satisfies C, and $C(t) = \beta$, where $\beta \in R^-$, which is a negative real number $\mathcal{R}^-$, is the value of the evaluation if t violates C.*

The value α in Definition 1 represents the *reward* granted to a tuple t for satisfying a constraint C. A higher value for the reward α denotes greater satisfaction. The value β represents the *penalty* imposed on a tuple t for violating the

constraint. A lower negative value for α denotes a greater degree of violation. Observe that in Definition 1, we try to capture the intuitive idea behind a reward and a penalty by letting the reward be a non-negative real number (rewards are positive) and the penalty be a negative real number (penalties are negative).

Designers should make domain decisions about the amount of a reward α and a penalty β. For instance, in a car purchase domain, designers may give a greater reward for newer cars. Therefore, they may define the evaluation for a constraint on a year in which a car was made such as "a 2000 or later" as $\geq\!(y, 2000) = y - 2000$. Observe that a 2001 car has a reward of 1 and a 2002 car has a reward of 2, which means that a 2002 car has a greater satisfiability degree according to this evaluation. Also observe that a 1999 car has a penalty of -1 and a 1980 car has a penalty of -20, which means that a 1999 car has much less of a penalty than a 1980 car.

An evaluation function can also impose a fixed penalty when ordering between values is not obvious. As an example, a constraint of the form "Brand = Canon" on digital camera brands can be defined as

$$BrandEqual(x, \text{“}Canon\text{”}) = \begin{cases} 0, & \text{if } x = \text{“}Canon\text{”}; \\ -1, & \text{otherwise} \end{cases}$$

We imposed a fixed penalty for any brand other than "Cannon", as Arnal suggested [Arn02], because it is not obvious how we can order penalties between brands other than "Cannon".

For equality constraints over which a penalty ordering is possible, designers can declare penalties. For instance, a designer may choose the evaluation for *EqualAppointmentTime(t, 10:00 AM)* to be $-(f(t) - f(10{:}00\ AM))^2$, where f is a function that converts a time to a unitless number. For example, the time 2:15 PM, which is the military time 14:15, could be converted to the integer 1415. For illustration purposes, we have assumed that the designer has chosen to square the difference to give proportionally less of a penalty to times close to 10:00 AM.

3 Underconstrained Service Requests

Underconstrained service requests admit too many solutions. In this section, we discuss two ways to provide users with the best-m solutions out of n solutions. First, we propose an expectation-based elicitation process to elicit additional constraints and apply them to solutions. Applying additional constraints to solutions may reduce the number of solutions and may also make the resulting solutions more desirable [SL01,FPTV04]. Second, we propose an ordering over solutions based on our extension for constraints in Definition 1 along with Pareto optimality based on this ordering to select the best-m solutions.

3.1 Constraint Elicitation Using Expectations

We associate expectations with concepts of a domain ontology. An *expectation* is the probability that value(s) for a concept appear in a service request. The

expectation is, therefore, a number in the interval [0, 1], where the low and high extremes of the interval mean, respectively, that a value for the concept is not and is certainly expected to appear in a service request. Values in the open interval (0, 1) represent varying degrees of expectations.

Domain ontology designers estimate the expectations associated with concepts. Although there may be several ways to estimate the expectations, we suggest two general ways. First, designers can estimate the expectation using their knowledge of the domain. Second, designers can analyze service requests in the domain of the ontology and count the frequency of appearance for each concept in the domain ontology. Further, this latter method leads to the possibility that the expectations can be adjusted as the system runs.

Unlike other approaches to constraint elicitation (e.g. [LHL97,SL01,PFK03]), which are built on an assumption that users can impose additional constraints if they review some examples of solutions, we let the resolution process take the initiative and suggest the concepts on which to impose constraints according to the associated expectations with these concepts. The intuitive idea is that the resolution process can order the concepts based on their associated expectations and make reasonable suggestions to users to constrain concept values, starting from the concept associated with the highest expectation for which there is, as of yet, no constraint.

The elicitation process terminates when one of the following three conditions holds. First, the most recent elicited constraint is unsatisfiable in which case the service request becomes overconstrained and the resolution process uses the techniques in Section 4 to handle this situation. Second, the solution space is reduced to m or fewer solutions, in which case the system offers these solutions to users to evaluate and choose one. Third, there is no other concept in the ordering of concepts associated with an expectation that exceeds a prespecified threshold.

To demonstrate the idea of constraint elicitation using expectations, note that the car purchase request in Section 1 does not specify a constraint on the model of the car. Assuming that the expectation associated with *Model*, say 0.6, is the highest among the unconstrained concepts and is above the threshold, say 0.5, the resolution process suggests that the user could impose a constraint on the model. If a user wishes to constrain *Model* to be "Stratus" the resolution process can restrict the solutions to Dodge Stratuses.

3.2 Selecting the Best-m Solutions

Our extension to the binary view of constraints (Definition 1) provides a way to impose an ordering over solutions based on rewards granted to each solution for satisfying the service request constraints. Let $S = \{s_1, ..., s_n\}$ be a set of solutions each of which satisfies every constraint in the set of constraints $C = \{C_1, ..., C_k\}$, which are imposed on a service request. The evaluation of the set of constraints C for a solution $s_i \in S$ returns a set of real numbers $\{C_1(s_i), ..., C_k(s_i)\}$, which are the rewards granted to s_i for satisfying the constraints.

Before computing an aggregate reward for a solution s_i over all constraints in C, we first divide each reward $C_j(s_i),\ 1 \leq j \leq k$, by $\underset{1 \leq i \leq n}{max} C_j(s_i)$, the maximum reward value over all solutions for constraint C_j. This normalizes the rewards to the interval [0, 1]. The purpose of the normalization is to discard the relative effects of large magnitude rewards across different constraints and thus to make it unnecessary to correlate values across different constraints. Let us denote the set $\{C_1(s_i), ..., C_k(s_i)\}$ after doing the normalization by $C^* = \{C_1^*(s_i), ..., C_k^*(s_i)\}$. Researchers have suggested several ways to compute combined evaluations (see [MA04] for a thorough survey). We linearly combine rewards in C^* yielding a combined reward ρ for a solution s_i as follows:

$$\rho_{C^*}(s_i) = \sum_{j=1}^{k} C_j^*(s_i); for\ i = 1, ..., n.$$

Definition 2. *Let s_i and s_j be two solutions and $C = \{C_1, ..., C_k\}$ be a set of constraints. We say that s_i is better than or equivalent to s_j, $s_i \succeq_\rho s_j$, with respect to C if $\rho_{C^*}(s_i) \geq \rho_{C^*}(s_j)$.*

To demonstrate the idea of reward-based ordering, let us suppose that we have a set of constraints $C = \{\leq$*(mileage, "30,000 miles")*, $\leq$*(price, "\$20,000")*$\}$ and two solutions s_1 = {mileage = "29,000 miles", price = "\$19,000"} and s_2 = {mileage = "29,900 miles", price = "\$18,000"}, then designers might decide to grant a reward of 1000 for s_1 and of 100 for s_2 for satisfying the mileage constraint, and a reward of 1000 for s_1 and a reward of 2000 for s_2 for satisfying the price constraint. Given these rewards, we can normalize them to [0, 1] by dividing the mileage rewards by 1000 and the price rewards by 2000, yielding the normalized rewards 1 and 0.1 for s_1 and s_2 respectively for satisfying the mileage constraint and the normalized rewards 0.5 and 1 for s_1 and s_2 respectively for satisfying the price constraint. Based on Definition 2, $s_1 \succeq_\rho s_2$ because $\rho_{C^*}(s_1)$ = 1.5 and $\rho_{C^*}(s_2) = 1.1$.

The ordering $\succeq_\rho$ sorts the solutions according to their combined rewards from the solution with the highest combined reward to the lowest. (Any solutions with identical rewards appear in a random order within their own equality group.) Although this ordering does sort the solutions, it does not necessarily imply that the first m solutions are the best-m solutions. The sorting procedure considers only the combined rewards, but does not consider the rewards granted to the solutions for satisfying each individual constraint. The rewards of the individual constraints, C_1, ..., C_k, in C provide additional knowledge to differentiate among solutions based on Pareto optimality, which divides solutions into dominating and dominated solutions based a dominance relation.

Definition 3. *Let $C = \{C_1, ..., C_k\}$ be a set of constraints and $S = \{s_1, s_2, ..., s_n\}$ be a set of solutions. Let s_i, $s_j \in S$ be any two distinct solutions, we say that s_i dominates s_j if $\forall_{p \in \{1,\ ...,\ k\}}(C_p(s_i) \geq C_p(s_j))$ and $\exists_{q \in \{1,\ ...,\ k\}}(C_q(s_i) > C_q(s_j))$.*

Definition 3 says that the solution s_i, which dominates s_j, has rewards from all the constraints that are at least equal to the rewards for s_j and for at least

one of the constraints s_i has a strictly higher reward. Observe the that Definition 3 does not explicitly consider the combined reward $\rho_{C^*}(s_k)$. However, the combined reward is implicit in this definition in the sense that a solution can never dominate another solution with a higher combined reward.

Definition 3 provides the basis for our variation of Pareto optimality, a concept which Pareto defined over a century ago [Par97].

Definition 4. *Let $S = \{s_1, s_2, ..., s_n\}$ be a set of solutions for a service request. A solution $s_i \in S$ is said to be Pareto optimal if there does not exist an $s_j \in S$ such that s_j dominates s_i.*

The key idea in Definition 4 is that a solution cannot be Pareto optimal if it is dominated by another solution.

3.3 Resolution of Underconstrained Requests

To demonstrate our resolution procedure, consider our request for a Dodge (in the introduction). The system first uses expectations to elicit additional constraints to reduce the number of solutions. Since the request does not constrain the model of the car and the expectation associated with the model is the highest among all the unconstrained concepts, the system suggests that the user constrains the model. Adding the constraint that the model be a "Stratus" drops the number of solutions to 53, which is still too many. Since there are no more concepts with an expectation higher than the threshold, 0.5, the system uses the ordering $\succeq_\rho$ and Pareto optimality to return the best-m solutions. Figure 4 shows the top 12 solutions ordered in ascending order based on their combined rewards $\rho_{C^*}(s_i)$. The rightmost column in Figure 4 shows whether a solution is Pareto optimal (✓) or not (×). For instance, the solution s_3 is not Pareto optimal because s_1 dominates it—s_1 is cheaper and has a lower mileage, although both have the same year. Since we have chosen $m = 5$, the system returns the first five Pareto optimal solutions, s_1, s_2, s_5, s_7, and s_{12}.

4 Overconstrained Service Requests

Overconstrained service requests admit no solution. As in Section 3, we discuss two ways to provide the best-m near solutions. First, we propose an ordering over near solutions and use it along with Pareto optimality to offer the best-m near solutions. Second, we propose an expectation-based relaxation process that suggests unsatisfied constraints for a user to relax.

4.1 Ordering Near Solutions

We combine the penalties and rewards, if any, of each near solution, and order the near solutions according to their combined penalties and rewards. Let $S = \{s_1, ..., s_n\}$ be a set of near solutions each of which violates one or more constraints from

Solution	Make	Model	Price	Year	Mileage	$\rho_{C^*}(s_i)$	Pareto Optimal
s_1	Dodge	Stratus	13,999.00	2005	15,775	2.499	✓
s_2	Dodge	Stratus	11,998.00	2004	23,404	2.497	✓
s_3	Dodge	Stratus	14,200.00	2005	16,008	2.476	×
s_4	Dodge	Stratus	14,557.00	2005	16,954	2.431	×
s_5	Dodge	Stratus	10,590.00	2003	38,608	2.360	✓
s_6	Dodge	Stratus	14,253.00	2004	17,457	2.332	×
s_7	Dodge	Stratus	10,987.00	2004	56,377	2.267	✓
s_8	Dodge	Stratus	13,999.00	2004	30,038	2.230	×
s_9	Dodge	Stratus	12,995.00	2004	40,477	2.226	×
s_{10}	Dodge	Stratus	12,577.00	2003	33,163	2.216	×
s_{11}	Dodge	Stratus	14,620.00	2004	32,406	2.149	×
s_{12}	Dodge	Stratus	8,975.00	2003	75,689	2.140	✓

Fig. 4. Solutions for the car purchase request.

a set of constraints $C = \{C_1, ..., C_k\}$. The evaluation of a set of constraints C for a near solution $s_i \in S$ returns a set of real numbers $\{C_1(s_i), ..., C_k(s_i)\}$, where each $C_k(s_i)$ is either a reward or a penalty. We divide these real numbers $C_j(s_i),\ 1 \leq j \leq k$ by $max_{1 \leq i \leq n} |C_j(s_i)|$, the maximum absolute reward or penalty value over all near solutions for constraint C_j. This normalizes the rewards and penalties to the interval [-1, 1]. Let us denote the set $\{C_1(s_i), ..., C_k(s_i)\}$ after normalization by $C^* = \{C_1^*(s_i), ..., C_k^*(s_i)\}$. We combine each $C_j^*(s_i)$ in C^* linearly, as we did in Section 3, yielding a combined penalty/reward ϕ for each near solution s_i as follows:

$$\phi_{C^*}(s_i) = \sum_{j=1}^{k} C_j^*(s_i); for\ i = 1, ..., n.$$

Greater values of $\phi_{C^*}(s_i)$ indicate lower penalties on s_i and (possibly) higher rewards. Thus, a high value of $\phi_{C^*}(s_i)$ denotes a better near solution s_i.

Definition 5. *Let s_i and s_j be two distinct near solutions and $C = \{C_1, ..., C_k\}$ be a set of constraints. We say that s_i is better than or equivalent to s_j, $s_i \succeq_\phi s_j$, with respect to C if $\phi_{C^*}(s_i) \geq \phi_{C^*}(s_j)$.*

We define a dominance relation and Pareto optimality based on the ordering $\succeq_\phi$ in Definition 5 in the same way as we defined them in Definitions 3 and 4.

4.2 Constraint Relaxation Using Expectations

For constraint relaxation we use the same expectation values for constraints as discussed in Subsection 3.1, but consider the lowest expectation values, rather than the highest, to be the candidates for relaxation. In addition, we consider the violation degree when we suggest constraints for relaxation. For instance, it is likely to be better to suggest relaxing a time constraint violated by 10 minutes than to suggest relaxing a distance constraint violated by 50 miles even though a distance constraint is likely to be associated with a lower expectation

value. Further, since we should not badger the user with questions, the number of suggested unsatisfied constraints should not exceed a prespecified threshold. Taking all these ideas into consideration, the system selects the constraints to suggest for relaxation based on the following procedure.

1. To avoid overloading the user with suggestions, select only near solutions that violate fewer constraints than a prespecified threshold.
2. To take the expectation values into account, compute the cost of the relaxation for each near solution based on the expectation using the equation $r(s_i) = \sum_k e_k C_k^*(s_i)$, where e_k is the expectation value associated with the constraint C_k and $C_k^*(s_i)$ is the normalized penalty imposed on s_i for C_k.
3. To take the overall degree of violation into account, select the near solution s_i with the lowest absolute value of $r(s_i)$ and suggest relaxing the constraints that s_i violates only to the degree necessary to satisfy the constraints of s_i.

We give an example in the next subsection.

4.3 Resolution of Overconstrained Requests

To demonstrate our resolution procedure, consider our request for an appointment (in the introduction). Figure 5 shows 8 near solutions for the request ordered in ascending order based on the combined penalty/reward $\phi_{C^*}(s_i)$, which appears in the second column from the right. The system tries first to suggest some constraints to relax using the expectations associated with the constraints. Figure 6 shows the constraints along with their associated expectation values and their rewards/penalties for each near solution. The rightmost column in Figure 6 shows the computed relaxation cost $r(s_i)$ for each near solution. Based on our relaxation procedure, the system could consider the near solution s_4 for suggesting relaxation because it has the lowest relaxation cost $r(s_i)$. The system does not, however, because s_4 violates three constraints, which exceeds the threshold we set, namely fewer than three constraints. The near solution s_3 satisfies our procedure requirements in the sense that s_3 violates two constraints and has the next lowest relaxation cost $r(s_i)$. The system therefore suggests letting the time be 12:40 PM instead of 1:00 PM and letting the date be the 19th instead of the 20th. If the user accepts these relaxed constraints, the system can offer s_3 as the best solution.

For the sake of further discussing the possibilities, we assume that the user does not accept the suggestion to relax the time and date constraints. To compute the best-m near solutions, the system sorts the near solutions based on the combined penalty/reward $\phi_{C^*}(s_i)$ and discards the dominated near solutions using the rewards and penalties information in Figure 6, i.e. $\phi_{C^*}(s_1) = -0.160 = -0.076 + 0.167 - 0.250$; $\phi_{C^*}(s_2) = -0.180 = -0.090 + 0.160 - 0.250$; and so forth. The rightmost column in Figure 5 shows whether a near solution s_i is Pareto optimal (✓) or not (×). Since $m = 5$, the system returns the first 5 Pareto optimal near solutions, which in our example are s_1, s_3, s_4, s_6, and s_8.

A closer look at the results in Figures 5 and 6 reveals that the returned near solutions are better than the ones filtered out. For instance, comparing the near

Near Solution	Insurance	Distance	Time	Date	$\phi_{C_*}(s_i)$	Pareto Optimal
s_1	IHC	16	1:00 PM	the 19th	−0.160	✓
s_2	IHC	18	1:10 PM	the 19th	−0.180	×
s_3	IHC	4	12:40 PM	the 19th	−0.257	✓
s_4	IHC	6	12:50 PM	the 19th	−0.264	✓
s_5	IHC	20	3:00 PM	the 19th	−0.271	×
s_6	IHC	8	1:40 PM	the 18th	−0.382	✓
s_7	IHC	18	2:20 PM	the 22nd	−0.479	×
s_8	IHC	3	11:30 AM	the 16th	−1.049	✓

Fig. 5. Near solutions for the appointment request.

	Insurance="IHC" Expectation=0.4	Distance≤ 5 Expectation=0.3	Time≥("1:00 PM") Expectation=0.8	Date="the 20th" Expectation=0.9	$r(s_i)$
s_1	0.000	−0.076	0.167	−0.250	−0.248
s_2	0.000	−0.090	0.160	−0.250	−0.252
s_3	0.000	0.007	−0.014	−0.250	−0.236
s_4	0.000	−0.007	−0.007	−0.250	−0.233
s_5	0.000	−0.102	0.083	−0.250	−0.256
s_6	0.000	−0.021	0.139	−0.500	−0.456
s_7	0.000	−0.090	0.111	−0.500	−0.477
s_8	0.000	0.014	−0.062	−1.000	−0.950

Fig. 6. Rewards and penalties for the near solutions.

solution s_1 to the discarded near solution s_2, we find that although both violate the date constraint to the same degree and satisfy the time constraint, s_1 violates the distance constraint less than s_2 and is closer to the requested time, 1:00 PM. Therefore, from the Pareto-optimality's viewpoint, given s_1 as a possibility, no user is likely to accept the near solution s_2.

5 Performance Analysis

To evaluate the performance of our system, we conducted a user study. The goal was to test whether there is a statistically significant difference between human choices and system choices. The subjects in our study were from both genders and from different academic disciplines and education levels—professors, graduate students, and undergraduate students at Brigham Young University. We gave every subject a request from a car purchase domain along with 32 cars that each satisfies all the constraints of the request, and another request from an appointment scheduling domain along with 19 near solutions that each satisfies some but not all the constraints of the appointment request. All the solutions and near solutions were randomly shuffled so as not to provide the subjects with any ordering information. We asked each subject to select and order the best-5 solutions out of 32 solutions for cars and the best-5 near solutions out of 19 near solutions for appointments.

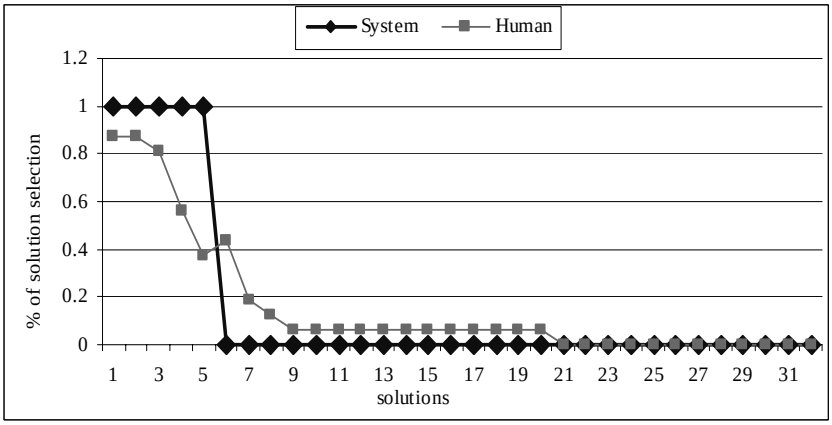

Fig. 7. Human solution selection compared to system solution selection.

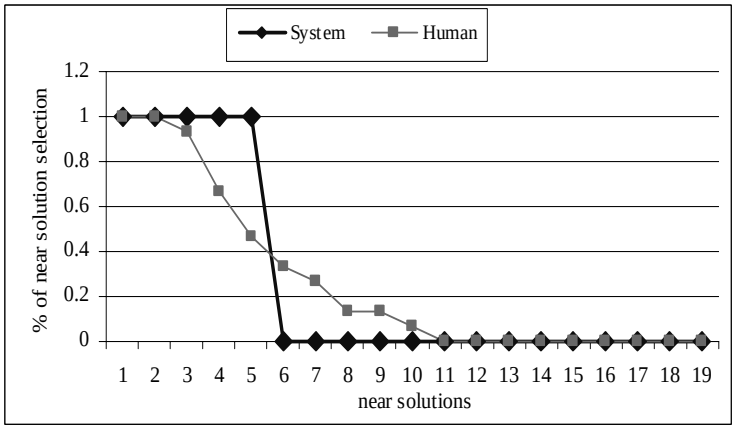

Fig. 8. Human near solution selection compared to system near solution selection.

To visualize the degree of agreement between system choices and human choices, we counted the number of times each solution was chosen by the 16 subjects for the car experiment and the number of times each near solution was chosen by the 15 subjects for the appointment experiment.[2] Figures 7 and 8 show the percentage of human subjects who chose each solution or near solution respectively. The first five solutions and near solutions are the ordered best-5 Pareto optimal solutions and near solutions. The remaining solutions and near solutions are ordered by decreasing percentage of selection by human subjects. As the figures show there is a high degree of agreement between the system's choices and the human subjects' choices. As Figure 7 shows, over 87% of the

[2] One of the subjects did not make choices for the appointment experiment.

System / Human	The best-5 solutions	Not the best-5 solutions	Total
The best-5 solutions	56	24	80
Not the best-5 solutions	24	408	432
Total	80	432	512

Fig. 9. Human versus system choices for the car experiment.

System / Human	The best-5 near solutions	Not the best-5 near solutions	Total
The best-5 near solutions	61	14	75
Not the best-5 near solutions	14	196	210
Total	75	210	285

Fig. 10. Human versus system choices for the appointment experiment.

subjects chose solutions 1 and 2, and over 81% of the subjects chose solution 3. Figure 8 shows an even higher degree of agreement. All the subjects chose near solutions 1 and 2, and over 96% of them chose near solution 3. The solutions and near solutions that were not chosen by the system as being among the best-5 were also selected less often by our human subjects, with the exception of solution 6 in Figure 7, which was selected by 43% of the human subjects, and the near solutions 6 and 7, which were chosen by the 33% and 26% of the human subjects. Interestingly, the system chose both solution 6 and near solution 6 as the 6^{th} Pareto optimal solution and near solution. Near solution 7, however, is not Pareto optimal. All the other solutions and near solutions were chosen by 20% or fewer of the subjects. Figures 7 and 8 reveal a definite pattern: human subjects chose a high percentage of the best-5 choices and a low percentage for choices not among the best-5 system choices.

To statistically measure the degree of agreement between system choices and human subjects choices, we ran an inter-observer agreement test [LK77] using the MINITAB 14 software package [min05]. The inter-observer agreement per observer pair (system and human) was determined with respect to the dichotomy: the best-5 solutions or the best-5 near solutions and not the best-5. Figures 9 and 10 show the distribution of agreement and disagreement between the system and our human subjects. We disregarded the order in which each subject ordered the best-5 solutions or near solutions, and tallied the number of solutions and near solutions chosen by subjects that belong to the best-5 solutions and the best-5 near solutions selected by the system. We also tallied the number of solutions and near solutions that were not chosen by the system and the subjects as the best-5. For instance, the 16 subjects for the car experiment made 80 choices of which 56 belong to the best-5 system choices and 24 do not. Further, of the 432 solutions not chosen, 24 were among the best-5 system choices while 408 were also not chosen by the system. Figure 11 shows the statistical summary for the car and appointment experiments. The overall agreement, P_o, and the agree-

Agreement index	Type of agreement	Car experiment	Appointment experiment
P_o	overall	0.91	0.90
P_{pos}	the best-5	0.70	0.81
P_{neg}	not the best-5	0.94	0.93
P_e	due to chance	0.73	0.61
Cohen kappa κ	chance corrected	0.67	0.74
95% Confidence interval for κ		[0.58, 0.76]	[0.65, 0.83]

Fig. 11. Statistical summary.

ment due to chance, P_e, for the car experiment are 0.91 and 0.73 respectively with a Cohen kappa κ value of 0.67, and for the appointment experiment are 0.90 and 0.61 with a κ value of 0.74. Based on the Landis-Koch interpretation for κ values [LK77], the two κ values indicate "substantial" agreement between the system and the subjects. The 95% confidence intervals for κ in Figure 11, however, indicate that the agreement may range from "moderate" (0.58) to "substantial" (0.76) for the car experiment and from "substantial" (0.65) to "almost perfect" (0.83) for the appointment experiment. It is useful also, as suggested in [CF90], to compute two more indices, namely the positive agreement P_{pos} on the best-5 and the negative agreement P_{neg} on those not among the best-5. The positive agreement, P_{pos}, for the car experiment and for the appointment experiment were respectively 0.70 and 0.81 whereas the negative agreement, P_{neg}, were respectively 0.94 and 0.93. All these numbers show a high agreement between the system and human subjects on both the best-5 and not among the best-5 (near) solutions. We next considered how the system and each subject ordered the best-5 solutions and near solutions. The κ values for the car experiment was 0.43 and for the appointment experiment was 0.61, indicating respectively "moderate" and "substantial" agreement between system ordering and subject ordering for the best-5 solutions and the best-5 near solutions.

6 Conclusions and Future Work

We proposed techniques to handle underconstrained and overconstrained systems of conjunctive constraints for service requests. These techniques depend on defining an ordering over the solutions or near solutions along with Pareto optimality to discard dominated solutions or near solutions. From among the ordered Pareto optimal solutions or near solutions, we select the best-m. We also introduced expectation values as domain knowledge and proposed an expectation-based process to elicit or relax constraints respectively for underconstrained and overconstrained requests. We conducted experiments to test our proposed ordering and Pareto optimality techniques and found substantial agreement between the system and human behavior.

Although still preliminary, the results are promising. As future work, we plan to do more user studies on additional domains with a larger number of subjects. In addition, we need to develop a dialog generation system for user interaction

and to conduct a field test for the generated dialog. Finally, we should integrate our resolution techniques into a service request architecture, such as the semantic web.

Acknowledgements

We appreciate Del T. Scott from the Department of Statistics, Brigham Young University, for his help with our statistical analysis. We also appreciate the help of all the subjects who participated in the experiments.

References

[AMEL05] M. J. Al-Muhammed, D. W. Embley, and S. W. Liddle. Conceptual Model Based Semantic Web Services. In *Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005)*, pages 288–303, Klagenfurt, Austria, October 2005.

[Arn02] M. T. Arnal. *Scalable Intelligent Electronic Catalogs.* PhD Dissertation, Swiss Federal Institute of Technology in Lausanne (EPFL), 2002.

[CF90] D. Cicchetti and A. Feinstein. High Agreement But Low Kappa. II. Resolving The Paradoxes. *Journal of Clinical Epidemiology*, 43(6):551–558, 1990.

[Fel80] A. M. Feldman. *Welfare Economics and Social Choice Theory.* Kluwer, Boston, 1980.

[FPTV04] B. Faltings, P. Pu, M. Torrens, and P. Viappiani. Designing Example-Critiquing Interaction. In *Proceedings of the 9th International Conference on Intelligent User Interface*, pages 22–29, Funchal, Portugal, November 2004.

[LHL97] G. Linden, S. Hanks, and N. Lesh. Interactive Assesment of User Preference Models: The Automated Travel Assistant. In *Proceedings of the 6th International Conference on User Modeling (UM97)*, pages 67–78, Vienna, New York, June 1997.

[LK77] J. R. Landis and G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174, 1977.

[MA04] R. T. Marler and J. S. Arora. Survey of Multi-Objective Optimization Methods for Engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

[min05] Minitab 14.2 Statitiscal Software. Website, 2005. www.minitab.com.

[Par97] V. Pareto. *Cours d'économie politique.* F. Rouge, Lausanne, Switzerland, 1897.

[PFK03] P. Pu, B. Faltings, and P. Kumar. User-Involved Tradeoff Analysis in Configuration Tasks. In *Proceedings of the 3rd International Workshop on User-Interaction in Constraint Satisfaction*, pages 85–102, Kinsale, Ireland, September 2003.

[SL01] S. Shearin and H. Lieberman. Intelligent Profiling by Example. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, pages 145–151, Santa Fe, New Mexico, January 2001.