Uniform Framework for Solving Over-Constrained and Optimization Problems

Hana Rudová, Luděk Matyska

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno
Czech Republic
tel: +420-5-41512343
e-mail: {hanka,ludek}@fi.muni.cz

September 22, 1999

Abstract

Variables' annotations as a new uniform framework for solving over-constrained and optimization problems are introduced. Interpretation of annotations representing preferences of variables in particular constraints is presented. Correspondence with constraint hierarchies and possibilistic CSPs allows solving of over-constrained problems. First steps in optimization area lead towards computation of variable ordering based on annotations. Real university timetabling problem is described and solved with annotations.

Keywords: Preferences of Variables, Over-Constrained Problems, Optimization Problems, Timetabling Application.

1 Introduction

Preferences in over-constrained problems may be defined with respect to importance or meaning of variable's occurence in constraint and such preferences may be represented through annotations. Annotation as a local preference of variable in constraint may also serve as a source for computing variable ordering. Annotations are stated as a part of the definition of a problem, separately and independ on the way it is resolved. Subsequently they may be used in the resolution process regardless of its kind.

Variables' annotations could be suitable for application areas such as planning or scheduling. Let us consider the timetabling problem stated in variables which represent teachers (dean, professors, assistants, ...), rooms (small, medium sized, large, ...), and courses (more and less engaged by students). These various degrees of importance define annotations which may be stated directly as a part of input data. If a problem has too large number of solutions to explore them all then annotations are used to generate more interesting solutions as soon as possible and when the problems becomes over-constrained annotations can define preferences for selection feasible solution of the problem.

Another interesting example is the problem of the search for the optimal sequence of aircraft departures from a runway. Because practically all the constraints are safety regulations, we can only change aircraft allocation to different time slots. In an over-constrained situation the only allowed action is the removal of an aircraft from further consideration. We can assign preferences to individual variables (planes) and define such a comparator which simply abandons the aircraft with the smallest preferences. In standard constraint satisfaction the same preferences allow us to find for example solutions with more important aircrafts in preferred time slots prior to other solutions.

Various types of preferences, priorities, satisfaction degrees, or weights were proposed to find solutions of problems with uncertainties, ill-defined problems, optimization problems, and over-constrained problems where some kind of softness have to be involved to get feasible solutions. The most simple framework, the maximal constraint satisfaction (MAX-CSP) [FW92] seeks a solution that satisfies as many constraints as possible. Weighted constraint satisfaction considers weights for each constraint and minimizes weighted sum of unsatisfied constraints. Both of these systems are used for solving over-constrained problems. Possibilistic CSP [Sch92] assigns to each constraint some preference degree, which express necessity of its satisfaction. Fuzzy CSP [DFP96] considers constraint as a relation assigning to each tuple of values its level of preferences. Preference degrees in both approaches are combined with help of fuzzy sets, possibility theory and possibilistic logic. These approaches are suitable for solving ill-defined problems, problems with uncertainties, and partially inconsistent problems. Constraint hierarchies [BFBW92, WB93] define several levels of constraints. The violations of constraints are minimized level by level subsequently with help of weighting the importance of constraints. Partial constraint satisfaction [FW92] is a general approach which finds a new problem with minimal distance from original problem with help of some metrics. Valued CSP [SFV95] and Semiring-based CSP [BMR97, GC98] (compared in [BFM$^+$96]) define monoid and lattice structure, which specify type of used preferences and manipulation with them. They are very general frameworks for constraint satisfaction and optimization. Above mentioned approaches may be described as a specification of these two frameworks. An approach the closest to ours, the Variable Valued CSP was only briefly described in [LV97, VLS96] and applied for the daily management of an earth observation satellite. There, each variable is associated with some weight expressing its importance. The objective is to produce a partial assignment of the problem variables which satisfies all imperative constraints and maximizes the sum of the weights of the assigned variables.

Preferences of all these systems specify how to compute one or more, best or optimal solution. Unfortunately, the search space is often too large to be fully traversed and explored. In such cases some strategy is needed to deal with this situation, however the above mentioned systems don't offer any. The annotations were introduced to help solve such a situation as they suggest some route through the search space which could lead to at least sub-optimal solution.

2 Annotations' Properties

A constraint system with variables' annotations is composed from a set of variables V, a set of constraints C, and from the finite domain of variables D.

Annotations are defined by a triple $(\mathcal{A}, \preceq, \circledast)$

- $\mathcal{A}$ as a set of annotations;
- $\preceq$ as an ordering on $\mathcal{A}$;
  if $a, b \in \mathcal{A}$ then $a \preceq b$ means a is more preferred annotation than b;
- $\underset{a_i \in A}{\circledast} a_i$ (finite $A \subset \mathcal{A}$) as a function computing global annotation; it is defined by applying either $\circledast : \mathcal{A}^k \to \mathcal{A}$ or a commutative and associative binary operation closed on $\mathcal{A}$.

Annotations are local to a constraint, i.e., any variable may have different annotations in different constraints. Function $a : C \times V \to \mathcal{A}$ determines the annotation of every variable in every constraint.

Local variable's annotation is used for computing several characteristics of underlying constraint system, so called global annotations. Let us define $var(c) \subseteq V$ as a set of variables in the constraint c. Then we can define global variable annotation by combining variable's annotations in all constraints where variable occurs:

$$av : V \to \mathcal{A} \ , \ av(v) = \underset{\{c \in C \mid v \in var(c)\}}{\circledast} a(c, v) \ . \tag{1}$$

Constraint annotation is defined by combining all variables' annotations in given constraint:

$$ac : C \to \mathcal{A} \ , \ ac(c) = \underset{\{v \in var(c)\}}{\circledast} a(c, v) \ . \tag{2}$$

Global constraint annotation is given by combining global variable annotations for variables which occur in the constraint:

$$acv : C \to \mathcal{A} \ , \ acv(c) = \underset{\{v \in var(c)\}}{\circledast} av(v) \ . \tag{3}$$

Typical examples of $(\mathcal{A}, \preceq, \circledast)$ are intervals (for example $\langle 0, 1 \rangle$) over real numbers, ordering $\geq$ over real numbers, and sum or arithmetic average, resp.

3 Annotations in Over-Constrained Problems

The small timetabling example illustrates how annotations may be defined and applied for solving an over-constrained problem:

```
c1: DayLecture@strong + 1 #=< DayPractice@medium
c2: DayLecture@weak in 4..5
c3: DayPractice@weak in 1..4
```

Constraint c1 states that the practice should be preferably taught at least one day after the lecture. Preferences express here that the lecture (taught by a professor) is more preferred than the practice (taught by an assistant). Next the lecture has to be taught on Thursday or Friday (c2) and the practice from Monday to Thursday (c3). These constraints form a kind of hierarchy: the constraint c1 with the highest preferences must be satisfied first and then we may try to satisfy constraints c2 and c3. It is possible to satisfy c1 but not c2 and c3 taken together. The constraint c2 influences variable with higher annotations (look at c1), so this constraint is also satisfied. Then, trying to minimize the overall constraint violation, we get (a kind of) optimal solution {DayLecture=4, DayPractice=5}. By classical hierarchy where c1 is annotated by strong or medium, the solution {DayLecture=3, DayPractice=4} is also obtained. But this solution is not optimal from our point of view. The different requirements towards the lectures and practices must be stated by assigning different preferences to c2 and c3 and so these constraints must be ordered. But this ordering could be incorrect with respect to other constraints in a more complex problem. Also, the exact location of the two appropriate constraints need not be easy found in this context.

We have defined mappings of variables' annotations to different frameworks [Rud98b, Rud98c] solving over-constrained problems — to constraint hierarchies [BFBW92, WB93] and possibilistic CSPs [Sch92]. Particular mappings give us several interpretations of annotations and they may be used as examples of possible semantics of annotations in over-constrained problems.

3.1 Mapping to Possibilistic CSPs

The interpretation of annotations through possibilistic CSP emphasizes global constraint annotations whereas the importance of particular constraint annotation is deferred.

The specifications of the general definitions for this mapping are $\mathcal{A} = \langle 0, 1 \rangle$, ordering $\geq$ over real numbers as $\preceq$, and the arithmetic average over real numbers as $\circledast$.

Let us define an error function $e(c\theta)$ which indicates how nearly constraint c is satisfied for a valuation θ. This error function can be trivial ($e(c\theta) = 0/1$ means c is satisfied/unsatisfied) or we can define the error function with help of the domain's metric.

Every solution θ has the error

$$E(C\theta) = \max_{\{c \in C\}} acv(c)e(c\theta)$$

and the optimal solution is the solution with minimal error.

3.2 Mapping to Constraint Hierarchy

Constraint hierarchy framework assigns to every constraint some level of preference and constraint violations are minimized level by level subsequently. Our mapping constructs the hierarchy over constraint annotations ac, with additional order imposed by global constraint annotations acv within each level.

Basic annotation structure $(\mathcal{A}, \preceq, \circledast)$ may be specified as follows: interval $\langle 0, 10 \rangle$, standard ordering $\geq$, and the arithmetic average over real numbers for $\mathcal{A}$, $\preceq$, and $\circledast$, resp. Constraint hierarchy $C = C_0 \cup C_1 \cup \ldots \cup C_n$ is a union of disjoint sets C_i where preferences of constraints decrease with increasing value of i. The hierarchy is constructed over constraint annotations ac:

$$C_i = \{c \in C | \, (\forall d \in C_j, j < i : ac(d) > ac(c)) \wedge (\forall e \in C_k, i < k : ac(c) > ac(e))\} \ .$$

The level C_0 is a set of required constraints which have to be satisfied. The annotation of every variable in constraint at this level is equal to 10.

A solution of this system is a valuation $\theta : V \to D$ with minimal error vector

$$E(C\theta) = [E(C_1\theta), \ldots, E(C_n\theta)]$$

compared in lexicographic order. $E(C_i\theta)$ introduces an error of the level i. The specification of error $E(C_i\theta)$ depends on the type of chosen comparator. As an example we describe the error of one level for the mapping with weighted-sum-better $E_{wsb}(C_i\theta)$ and worst-case-better $E_{wcb}(C_i\theta)$ comparator[1], where the value of global constraint annotation $acv(c)$ is understood as a weight of constraint c:

$$E_{wsb}(C_i\theta) = \sum_{\{c \in C_i\}} acv(c)\, e(c\theta) \ \ , \ \ E_{wcb}(C_i\theta) = \max_{\{c \in C_i\}} acv(c) e(c\theta) \ \ .$$

An error function $e(c\theta)$ indicating how nearly constraint c is satisfied for a valuation θ is defined just as in Section 3.1.

Now we can go back to introductory timetabling example with variables' annotations. We solve it by the mapping to constraint hierarchy with weighted-sum-metric-better[2] comparator. The annotations are strong = 3, medium = 2, and weak = 1. We split constraints into constraint hierarchy using the corresponding constraint annotations ac: $ac(\mathsf{c1}) = \frac{3+2}{2}$, $ac(\mathsf{c2}) = 1$, $ac(\mathsf{c3}) = 1$. These constraint annotations imply following construction of the hierarchy $C = C_1 \cup C_2 = \{\mathsf{c1}\} \cup \{\mathsf{c2}, \mathsf{c3}\}$. Constraint c1 from the first level can be satisfied and the error of this level is equal to 0. Second and third constraints taken together can not be satisfied and we need to compute global constraint annotations for constraints c2 and c3: $acv(\mathsf{c2}) = av(\mathsf{DayLecture}) = 2$, $acv(\mathsf{c3}) = av(\mathsf{DayPractice}) = 3/2$. With respect to the smaller weight of c3, we get optimal solution $\theta = \{[\mathsf{DayLecture}, 4], [\mathsf{DayPractice}, 5]\}$ with error vector $E(C\theta) = [acv(\mathsf{c1}) \times e(\mathsf{c1}\theta), acv(\mathsf{c2}) \times e(\mathsf{c2}\theta) + acv(\mathsf{c3}) \times e(\mathsf{c3}\theta)] = [0, 0 + acv(\mathsf{c3}) \times 1] = [0, 1.5]$.

4 Optimization Problems

In optimization problems large search spaces have to be explored to find the best solution. In many cases different heuristics are used to find some sub-optimal solution because the whole space can not be explored due to its complexity. Annotations as a source for computing variable ordering may also help to find more interesting solutions as soon as possible, especially with help of value ordering heuristics.

The first intuitive approach to computing variable ordering leads through global variable annotations av which are computed for each variable by combining variable annotations in all constraints where variable occurs (Eqn. 1). During computation variables are instantiated in order given by value of global variable annotation av and ordering of annotations $\preceq$.

Another possibility is to consider quite different semantics of annotations which emphasizes the locality of variable annotations within individual constraints. Such annotations would define only ordering inside the constraints and complete variable ordering would be computed from these local partial orderings. These local orderings would probably exhibit higher degree of consistency when large amount of preferences have to be defined and used.

Our further ideas concern dynamic manipulation with annotations allowing to compute dynamic variable ordering — annotations should consider current state in the search space and help to navigate through it with respect to currently achieved results.

[1] Detailed description of mapping to constraint hierarchies with other comparators may be found in [Rud98a].
[2] We apply weighted-sum-better comparator with metric error function $e(c\theta)$.

5 Timetabling

We try to apply annotations for solving timetabling problem which is a typical application of constraint programming technology. Our university timetabling problem consists in assigning class rooms to courses and placing courses in time. Annotations currently represent preferences of teachers given by the seniority of each teacher and numbers of students pre-enrolled on the course (expressing interest on a given course). These annotations are combined with help of global variable annotation and sum as a combining function. Computed global variable annotations serve as a source for computing variable ordering in optimization problem.

Our timetabling problem is a bit more complex than usual, because we would like to create individual timetable for every student — as (s)he registers for a set of courses. Also the number of required courses is small, majority of courses are optional and so the sets of registered courses for each student can be very different. Still higher level of complexity adds another problem: each course may consists from lecture or seminar or lecture+seminar, where the number of lectures and/or seminars is determined with respect to the number of students pre-enrolled on a course. Each student should visit one lecture and/or one seminar for each registered course. Computing an ideal timetable should solve the problem of student's splitting into groups such that each student is able to visit lecture and/or seminar for each his/her specified course.

Currently we start to solve simplified version of this problem. We have generated some feasible division of students into groups and we wish to concentrate on the problem of minimal conflict among courses of each student with respect to given division of courses. With this we are able to compute for each generated timetable the total number of courses which overlap for any student in given division. This number gives us the worst case cost's estimation of generated timetable — some places in lectures/seminars are still free with respect to existing conflicts for some students and students are also allowed to swap lectures/seminars with other students to improve their timetables. We plan to apply this cost evaluation during exploration of the search space and to compare different approaches of the cost application.

Let us just shortly introduce basic types of implemented constraints:

- each teacher gives only one course at a time;
- only one course can take place in a class room at a time;
- each course may be taught in several class rooms of suitable size and type;
- each teacher specifies the maximal number of courses per day;
- one teacher can teach one course in two specific class rooms with help of camera and video projection at the same time.

We have implemented two versions — in SICStus Prolog[3] and in ILOG[4], an object oriented library in C++. With respect to growing size of our school we can use input data of different scope. Each instance is timetabled for 5 days with 13 teaching units. The smallest instance has about 250 courses, about 100 teachers and 12 class rooms.

[3]URL `http://www.sics.se/ps/sicstus.html`

[4]URL `http://www.ilog.co.uk`

Our problem still behaves like optimization problem with respect to missing constraints in implemented system. We expect that the requirements of individual teachers will introduce additional complexity to the timetable. Here the problem consists only in insufficient existing input data. We already started to work with people responsible for the timetable at our school for the next semester and so we look for better input data in a short term. The second possibility consists in a more comprehensive consideration of the problem of registration. But this problem is still open for the future because we would like to implement the above proposed solution first and also compare achieved result for system with and without annotations.

6 Conclusion

We proposed a new uniform approach for solving over-constrained and optimization problems using variables' annotations. We defined the mappings from variables' annotations to existing over-constrained systems which allows us to express and make sense of the semantics of variables' annotations in the background of existing systems. We presented mappings of variables' annotations to possibilistic CSP and constraint hierarchies. We have done first steps in applying annotations for solving optimization problems through computing variable ordering. We hope that annotations help to solve the problems with large search spaces where they may suggest a route which leads to at least feasible if not even optimal solution.

In the future we would like to study potential of annotations for solving optimization problems. We try to use local ordering of annotations in constraints to compute complete variable ordering and we will consider annotations as a source for computing dynamic variable ordering. Our next research will also deal possible semantics of annotations in the context of constraint logic programming.

We applied annotations for solving timetabling problem in our school. We started with optimization problem with annotations, we would like to add more constraints and widely apply annotations for solving this optimization problem which may soon become even over-constrained and then apply annotations to this over-constrained problem as well. We will take under consideration the cost estimation of generated timetable during computing of solution and we will also compare computed cost for system with and without annotations. We also intend to solve other problems from scheduling or planning application areas to show that annotations may help solve wide range of problems.

Acknowledgements

This work is supported by the Universities Development Fund of the Czech Republic under the contract # 407/1999.

References

[BFBW92] Alan Borning, Bjorn Freeman-Benson, and Molly Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, 1992.

[BFM+96] Stefano Bistarelli, Hélène Fargier, Ugo Montanari, Francesca Rossi, Thomas Schiex, and Gérard Verfaillie. Semiring-based CSPs and Valued CSPs: Basic properties and comparison. In Michael Jampel, Eugene Freuder, and Michael Maher, editors, *Over-Constrained Systems*, pages 111–150. Springer-Verlag LNCS 1106, August 1996.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint solving and optimization. *Journal of ACM*, 44(2):201–236, March 1997.

[DFP96] Didier Dubois, Hélène Fargier, and Henri Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 6:287–309, 1996.

[FW92] Eugene C. Freuder and Richard J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.

[GC98] Yan Georget and Philippe Codognet. Compiling Semiring-based constraints with `clp(FD,`S`)`. In Michael Maher and Jean-Franccois Puget, editors, *Principles and Practice of Constraint Programming – CP98*, pages 205–219. Springer-Verlag LNCS 1520, 1998.

[LV97] M. Lemaître and G. Verfaillie. Daily management of an earth observation satellite : comparison of ILOG Solver with dedicated algorithms for Valued Constraint Satisfaction Problems. In *Proceedings of Third ILOG International Users Meeting*, Paris, France, July 1997.

[Rud98a] Hana Rudová. Constraints with variables' annotations and constraint hierarchies. In Branislav Rovan, editor, *SOFSEM'98: Theory and Practice of Informatics*, pages 409–418. Springer-Verlag LNCS 1521, 1998.

[Rud98b] Hana Rudová. Constraints with variables' annotations. In Henri Prade, editor, *13th European Conference on Artificial Intelligence Proceedings*, pages 261–262. John Wiley & Sons, Ltd., 1998.

[Rud98c] Hana Rudová. Constraints with variables' annotations. Technical Report FIMU-RS-98-04, Faculty of Informatics Masaryk University, 1998. In http://www.fi.muni.cz/informatics/reports/.

[Sch92] Thomas Schiex. Possibilistic constraint satisfaction problems or "How to handle soft constraints ?". In *8th International Conference on Uncertainty in Artificial Intelligence*, pages 268–275, Stanford, CA, July 1992.

[SFV95] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 631–639, San Mateo, August 1995. Morgan Kaufmann.

[VLS96] G. Verfaillie, M. Lemaître, and T. Schiex. Russian doll search for solving constraint optimization problems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96) and Eighth Conference on Innovative Applications of Artificial Intelligence (IAAI-96)*, pages 181–187, Portland, OR, USA, 1996.

[WB93] Molly Wilson and Alan Borning. Hierarchical constraint logic programming. *Journal of Logic Programming*, 16(3,4):227–318, 1993.