Scribe Notes: 2/13/2013
Presenter: Tony Schnider
Scribe: Nate Stender
Topic: Soft Constraints (Ch. 9 of CP handbook)

Soft Constraints

Motivation

Soft constraints are used:

1. When we seek to find the best (optimal) solution to a problem,
2. When there are probabilistic constraints, or
3. When a problem is over-constrained (seek to satisfy as many constraints as possible).

In order to find the optimal solution, the entire search space must be searched. This problem is at least as hard a classical CSPs (a classical CSP can always be represented by one with soft constraints).

The speaker discussed two types of frameworks for soft constraints and two types of algorithms:

A. Specific frameworks: fuzzy & weighted
B. Local consistency
C. Generic frameworks: semiring-based & valued
D. Systematic search

A1. Fuzzy Sets

In usual set theory, any value of a set either belongs to the set or does not. In a *fuzzy* set, we ask "How much does this value belong to this set?" The value for the set can be 0: doesn't belong, 1: does belong, or between 0 and 1: for some degree of belonging.

A fuzzy constraint assigns every possible *tuple* in a relation a membership degree. The function that determines the degree of membership for a value/tuple is called a *membership function.* Thus, in fuzzy CSPs, the 'weights' are assigned to the individual tuples of a relation.

Fuzzy constraints are combined using *conjunctive combination* ($R_V \otimes R_W$). This combination amounts to taking the Cartesian product of the two relations choosing for a tuple the minimum value among the membership functions.

Question from Daniel G: Why do we want the minimum value when combining fuzzy constraints?

Answer: Because we want to find the "weakest link" to keep for the constraint. For example, if we want to maximize the happiness of all people, it is not fair to average because then some people would end up very happy, and some would end up very unhappy. We want to maximize the minimum happiness.

The satisfaction rate of an assignment is determined by the least satisfied constraint (the minimum membership degree). Any assignment with a non-zero satisfaction rate is considered a solution, and the solution with the maximum satisfaction rate is optimal.

Advantages:

- Good for applications where the system is only as good as its weakest link.
- A fuzzy CSP can fully represent a classical CSP
- Can change the comparison operator (from min) to represent other situations.

Disadvantages:

- We can have a tie between two "optimal solutions", even though one solution is better on specific constraints. (The decision is blinded by the minimum of both.)
- Solution, sort individual constraints by value and then break ties in that order.

A2. Weighted Constraints

- A *weighted constraint network* (WCSP) is a CSP with weighted constraints.
  - The cost of an assignment in the WCSP is the number of violated constraints.
  - The optimal solution with a complete assignment with minimal cost (and thus the least amount of violated constraints).
- k-weighted constraints are weighted constraints with the addition of an upper bound on the acceptable cost for a solution.
- By adding a limit, k-weighted CSPs can be tested for consistency, instead of just taking the best solution.

B. Local Consistency in Soft Constraints

Local consistency is more difficult in SCSPs than in (crisp) CSPs because a violated constraint does not indicate inconsistency. Thus, values can be pruned only when they are *node* inconsistent, not when they are arc inconsistent.

- A value is node consistent if it is below an upper limit T.
- A value is arc-consistent if it is node consistent, and it has at least one support in its neighboring variable s.t. the cost of the tuple in the constraint equals a lower limit $\perp$.

Question by Robert: Is there ever a time when $\perp$ is greater than 0?

Answer: Not sure, but for representation purposes you could always normalize $\perp$ to 0 by subtracting its value from each weight.

In order to enforce arc consistency, we push the costs of binary constraints onto the unary constraints. To do this, the minimum cost of the binary variable is added to the unary constraint of one of the participating variables and then subtracted from the binary constraint. Adding and subtracting to constraint weights has a special meaning in WCSPs:

$$a \ominus b = \begin{cases} a - b & : a \neq k \\ k & : a = k \end{cases} \qquad a \oplus b = \begin{cases} a + b & : a + b < k \\ k & : a + b \geq k \end{cases}$$

In this way, the cost of binary constraints is passed to the unary constraints, and thus node consistency can be used to enforce arc consistency. One problem that can arise from this method is that the order of evaluation (node pruning) may produce *different* structures in the CSP:

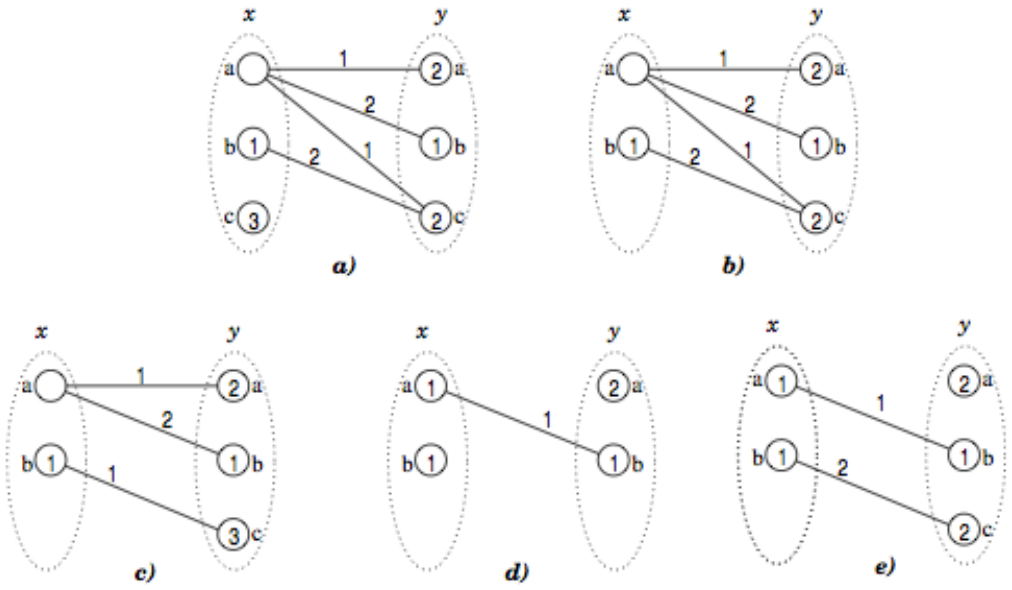

Figure 1: Example of difference CSP structures due to different orders of evaluation

Question by Nate: Are there heuristics for selecting the order of pruning?

Answer: None mentioned in the paper, but there could be work done on this topic since the paper was published.

W-AC3 and W-AC2001 algorithms are similar to their hard-CSP counterparts. They have an added step of making variables arc-consistent by forcing support for arc consistency. In addition, variables must be rechecked for node consistency whenever they are modified.

C. Generic Soft Constraint Frameworks

The goal of developing generic frameworks for SCSPs is to allow:

1. the development of generic algorithms that will work on all types of soft constraints, instead of having to create a new one for each type. And
2. the easy comparison between the frameworks.

For SCSPs, two main generic frameworks have been developed: semiring-based constraints and valued constraints.

Before we discuss semirings, I would like to introduce some terminology from algebra (answering instructor's question:

- ***A group (S,*)*** *is a set S with an operation * such that the operation * satisfies the following conditions:*
  - *Closure (x,y ∈S⇒ x*y∈S)*
  - *Associativity ((x*y)*z= x*(y*z))*
  - *Identity=idempotent (∃ e∈S, x*e=e*x=x)*
  - *Invertible (∀ x∈S ∃ y∈S s.t. x*y=y*x=e)*
- ***An Abelian group (S,*)*** *(or a commutative group) is a group that is also commutative (x*y=y*x).*
- ***A ring (S,+,*)*** *is a set S and two operations * and + such that*
  - *(S,+) is an Abelian group (+ is closed, associative, has identity element, invertible, and commutative)*
  - ** is associative ((x*y)*z= x*(y*z))*
  - ** distributes over + that is: x*(y+z)= (x*y)+(x*z)*
  - *Often, * is required to have an identity element*
- ***A semiring (S,+,*)*** *is a ring where*
  - *+ is not invertible*
  - ** has an annihilator*

C1. c-Semirings

A *c-Semiring* (*constraint* semiring) is a tuple $\langle E,+_s,\times_s,0,1\rangle$ with the following properties:

- E is a set of satisfaction degrees where $0\in E$, $1\in E$
- Operator $+_s$: is associative, commutative, and idempotent; closed in E; has 0 as a neutral element and 1 as an annihilator
- Operator $\times_s$: is associative and commutative; is closed in E; has 1 as a neutral element and 0 as an annihilator
- $\times_s$ distributes over $+_s$

c-Semiring operators

$\times_s$ is used to combine preferences

Let:

- R_C be completely satisfied
- R_U be completely unsatisfied
- R_l have satisfaction degree l
- $val_s(R)$ be the preference level of relation R

Then:

- $val_s(R_C) \times_s val_s(R_U) = 0$
- $val_s(R_C) \times_s val_s(R_l) = l$
- $val_s(R_l) \times_s val_s(R_l) = l$

$+_s$ is used to order preference levels

Let a and b be two preference levels:

- $a +_s b = a$ $\quad b \preccurlyeq_s a$
- $a +_s b = b$ $\quad a \preccurlyeq_s b$
- $a +_s b = c$ and $a \neq c$, $b \neq c$, incomparable

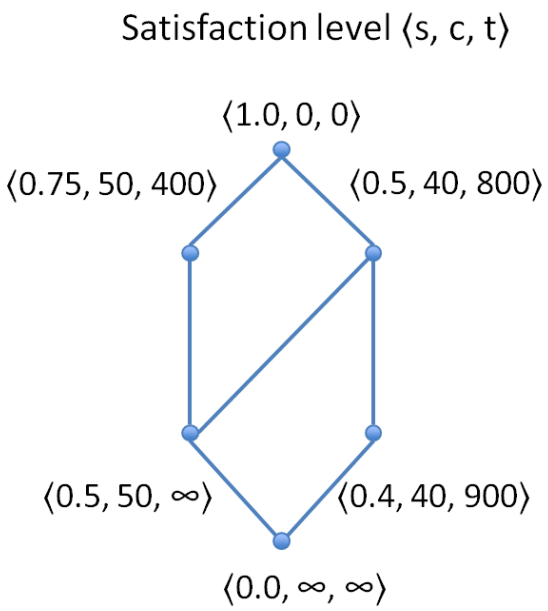

Figure 2: Hasse Diagram

$+_s$ Induces a *partial* ordering over preferences, specifically a lattice where $+_s$ is the least upper bound (LUB) of $\{a,b\}$.

c-Semirings can be partially ordered based on their level of satisfaction. Partial ordering allows multiple criteria to be optimized. Partial ordering can be represented using a Hasse diagram, as seen in Figure 2. In a Hasse diagram the higher a node is, the more satisfied it is.

A semiring constraint network is a tuple ⟨X,D,C,S⟩ where:

- X, D : same as in classical CSPs
- C is a set of soft constraints
- $S = \langle E, +_s, \times_s, 0, 1\rangle$ is a c-semiring

Questions Choueiry: what is the difference between c-semiring and semiring:

Answer: c stands for constraints

C2. Valued Constraints

Valued constraints are an alternative framework to semiring-based constraints that use a valuation structure instead of a semiring.

A valuation structure is a tuple $\langle E, \oplus, \preccurlyeq_v, \bot, \top \rangle$ where

- E is a set of values totally ordered by $\preccurlyeq_v$
- $\bot$ is a minimum element in E
- $\top$ is a maximum element in E
- $\oplus$ is an operator which
    - is associative, commutative, and monotonic
    - is closed in E
    - Has $\bot$ as a neutral element and $\top$ as an annihilator

Monotonicity induces total ordering: $a \preccurlyeq_v b \longrightarrow ((a \oplus c) \preccurlyeq_v (b \oplus c))$

Valuation structures are *totally* ordered, and thus they are less expressive than semirings, which can be partially ordered. Thus, a valuation structure can always be represented by a semiring. Alternatively, a semiring can only be represented by a valuation structure if the semiring is totally ordered.

C3. Operations on Soft Constraints

Soft constraints can be combined (or joined) without information loss. In a join, every possible tuple resulting from the combination is generated, and the membership function value is the minimum of the combined tuple values.

When soft constraints are projected over a variable (a variable is eliminated from the constraint) the membership function value is the maximum of the equivalent remaining tuples.

C4. Soft Constraint Solutions

Projecting the soft constraint over nothing, we get the maximum values, which is the network's consistency level.

As long as the preference level is above 0, then an assignment is complete (i.e., acceptable). The optimal solution is one s.t. no other assignment has a better consistency level.

Question Fikayo: Is it possible to have negative numbers in the tuples?

Answers: No, not directly, as this would break the minimum operation having 0 as the annihilator.

D. Search in Soft CSPs

There are two basic categories of search in CSPs.

1. Systematic search explores the entire search space and is guaranteed to find the optimal solution, but is usually too costly to perform.
2. Local search is more feasible because it is limited in resources (usually time), but it may not find the best solution.

Hybrid approaches can be used to take some benefit from both approaches.

A basic form of systematic search is depth-first branch-and-bound (DFBB). DFBB explores the search space in an DFS manner, while keeping track of an upper and lower bound. At any given level in the search, if a partial assignment's lower bound is greater than the upper bound (assuming a minimization task), search backtracks.

Most improvement on DFBB focuses on generating better lower-bound estimates. One such method is using partial forward checking (PFC) to generate a tighter bound by adding information from unassigned variables. PFC can be further improved by applying directional arc inconsistency (DAI) to include costs from completely unassigned constraints. One caveat to DAI is that it requires a static variable ordering.