Mini-bucket+
MPLP/SAC
Solving Min-sum problems

A finite COP is a triple $R = \langle X, D, F \rangle$ where:
- $X = \{X_1, \ldots, X_n\}$ - variables
- $D = \{D_1, \ldots, D_n\}$ - domains
- $F = \{f_1, \ldots, f_m\}$ - cost functions
- $S = \{S_1, \ldots, S_m\}$ - scopes

Global Cost Function

$F(X) = \sum_{i=1}^{m} f_i(X)$

$F^* = \min_x F(x)$

- **Exact** by Bucket elimination
- **Bounding scheme**: MBE, MPLP, Soft arc-consistency.
- **MBE**: capitalize on large cluster processed exactly. Similar to i-consistency for large $i$ (parameter $z$), directional, non-iterative.
- **SAC, MPLP**: equivalent to arc-consistency. Can be extended to cluster graphs, but may not be efficient. Based on cost shifting subject to equivalence-preserving transformations (EPT)
- **Goal**: MBE+MPLP/SAC can improve bounds of both in an efficient way.

$\mathbf{x} = (x_1, \ldots, x_n)$ to all the variables which maximizes the sum of the factors:

$\text{MAP}(\theta) = \max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\mathbf{x}_f).$ (1.1)
MBE

Let \( Q = \{Q_1, \ldots, Q_r\} \) partition of \( F \) into mini-buckets whose scope bounded by \( z \):

\[
f^* = \min \sum_x f_i(x_f) \geq \sum_{Q_j} \min \sum_{f \in Q_j} f(x_f)
\]
Mini-Bucket Approximation

Split a bucket into mini-buckets => bound complexity

bucket \((X) = \{ h_1, ..., h_r, h_{r+1}, ..., h_n \} \)

\[ h^X = \min_X \sum_{i=1}^n h_i \]

\( \{ h_1, ..., h_r \} \) \hspace{1cm} \( \{ h_{r+1}, ..., h_n \} \)

\[ g^X = \left( \min_X \sum_{i=1}^r h_i \right) + \left( \min_X \sum_{i=r+1}^n h_i \right) \]

\[ g^X \leq h^X \]

Exponential complexity decrease: \( O(e^n) \rightarrow O(e^r) + O(e^{n-r}) \)
Semantics of Mini-Bucket: Splitting a Node

Variables in different buckets are renamed and duplicated
(Kask et. al., 2001), (Geffner et. al., 2007), (Choi, Chavira, Darwiche, 2007),
(covering graphs, Ihler et. Al 2010.)

Before Splitting:
Network $N$

After Splitting:
Network $N'$
Optimal Soft Arc-consistency

- Given a set of functions $F = \{f_1, \ldots, f_r\}$ having scopes $S_1, \ldots, S_r$, over $n$ variables $X$, we compute functions on the same scopes $F' = \{f'_1, \ldots, f'_r\}$ and a set of unary functions $\{c_1, \ldots, c_n\}$ over single variables s.t:
  - $F \sim \{F', C\}$:
    $$\sum_{f \in F} f(x) = \sum_{i} c_i(x[i]) + \sum_{f' \in F'} f'(x)$$

- Where $f'$ is obtained by cost shifts:
  - $p_{fxi}$ is the cost shifted from $f$ (or scope($f$)) to the value $c_i(x_i)$ of $X_i$ in scope($f$).

$$\min F(x) \geq \sum_{i} \min_{x_i \in \text{scope}(f)} \sum_{f \in F} p_{fxi}(x_i) + \sum_{f \in F} \min_{x_i \in \text{scope}(f)} \{ (f(x_f) - \sum_{X_i \in \text{scope}(f)} p_{fxi}(x_i)) \} = Fp_f(x)$$

$$\min_x \sum_{f \in F} f(x_f) \geq \max_{p_f} Fp_f$$
Algorithm 1: Project and UnaryProject for soft arc and node consistency enforcing

Procedure Project(cS, i, a, α)
\[ c_i(a) \leftarrow c_i(a) \oplus \alpha; \]
foreach (t ∈ S) such that t[{i}] = a) do
\[ c_S(t) \leftarrow c_S(t) - \alpha; \]

Procedure UnaryProject(i, α)
\[ c_\emptyset \leftarrow c_\emptyset \oplus \alpha; \]
foreach (a ∈ di) do
\[ C_i(a) \leftarrow c_i(a) - \alpha; \]

A variable i is said to be node consistent [Larrosa, 2002]
if 1) for all values a ∈ di, ci(a) ⊕ c_\emptyset = k, 2) there exists a
value b ∈ di such that ci(b) = 0. A WCSP is node consistent
(NC) if every variable is node consistent

A value b ∈ dj is a support for a value a ∈ di along cij if cij(a, b) = 0.

Variable i is arc consistent if every value a ∈ di has a support in every
constraint cij ∈ C. A WCSP is arc consistent (AC) if every variable is arc and
node consistent.
Shifting Costs (cost compensation)

Subtract from source in order to preserve the problem

⇒ Equivalence Preserving Transformation
Equivalence Preserving Transformation

- Shifting costs from $f_{AB}$ to $A$
  
  \[
  \text{Shift}(f_{AB}, (A, w), 1)
  \]

- Can be reversed (e.g. $\text{Shift}(f_{AB}, (A, w), -1)$)

\[\begin{array}{c}
A \\
| 0 |
\end{array} \quad \begin{array}{c}
B \\
| 0 |
\end{array} \]

**Arc EPT:** shift cost in the scope of 1 cost function

Problem structure preserved
OSAC: The Optimal SAC: 
A continuous linear formulation

• \( u_A \): cost shifted from \( A \) to \( f_0 \)
• \( p_{Aa}^{AB} \): cost shifted from \( f_{AB} \) to \( (A,a) \)

\[
\max \sum u_i
\]

Subject to non negativity of costs

\[
\forall i \in X, \forall a \in d_i, \quad c_i(a) - u_i + \sum_{(c_S \in C'), (i \in S)} p_{i,a}^S \geq 0
\]

\[
\forall c_S \in C, |S| > 1, \forall t \in \ell(S) \quad c_S(t) - \sum_{i \in S} p_{i,t}\{i\} \geq 0
\]

\( n + m.r.d \) variables
\( n.d + m.d' \) linear constraints
Optimal Soft AC

• Solved by Linear Programming

• Polynomial time, rational costs (bounded arity \( r \))

• Computes an optimal set of EPT \((u_A, p_A^{AB})\) to apply simultaneously

• Stronger than AC, DAC, FDAC, EDAC...
  (or any local consistency that preserves scopes)
The OSAC algorithm
\[ P' = \text{OSAC}(P) \]

- Input \( P = (X,D,F = \{ f_1, \ldots, f_r \}) \)
- Output: \( Q = (X,D',F' = \{ f_1', \ldots, f_r', c_1, \ldots, c_n \}), P \sim Q \) s.t.
  \[ F \approx \max_{q \approx p} \sum_i \min_{x \in D_i} \{ c_i^q(x_i) \} \]

- Step 1: Apply LP(P) to obtain \( F' = \{ c_1, \ldots, c_n \}, \{ p_{fx_1}, \ldots, p_{fx_n} \} \)
  \( \forall x_i \forall f, \forall X_i \text{ scope}(f) \).
- Step 2: \( Q = (X,D',F' = \{ f_1', \ldots, f_r' \}) \) where \( f' \) are obtained by:
  \[ f_i'(t) = f_i(t) - \sum_{X_j \in \text{scope}(f_i)} p_{t[x_j]}^{f_i} \]
Combining MBE with OSAC

**MBE-OSAC(1)**

1. Process buckets from i=n to 1.
   - Partition into mini-bucket
   - \{mb_{i1},...,mb_{ij}\} \leftarrow sum functions in each mini-bucket
   - \{mb'_{i1},...,mb'_{ij}\} \leftarrow OSAC{mb_{i1},...,mb_{ij}}
   - \Lambda_{ij} \leftarrow \max_{\{X_{i}\}} mb'_{ij}

2. Return the sum of functions in the first bucket.

**Properties:**

1. The algorithm is exp(z) and solves LP problem in each bucket.
2. The output is a lower bound. But its quality may or may not improve over MBE *(question: any guarantee? In practice?)*
**mbosac(2):**
shift costs based on all bucket variables and then mini-bucket

**Input:** \( B_i = \{ f \in F \mid X_i \in \text{scope}(f) \} = \{ f_{i1}, \ldots, f_{it} \} \)

**Output:** messages \( \lambda_{i1}, \ldots, \lambda_{ir} \) placed in lower buckets based on their scope.

1. \( B_i^{osac} = \{ c_{i1}(x_{i1}), \ldots, c_{il}(x_{il}), f'_{i1}, \ldots, f'_{it} \} \leftarrow \text{OSAC}(f_{i1}, \ldots, f_{it}) \)
2. For \( j = 1 \) to \( r \) do (process each mini-bucket)

\[
\lambda'_{ij}(x) = \min_{X_i} \sum_{f \in Q_{ij}} f'
\]

End.

**Properties:** May not improve on BME. Will it improve over OSAC? In practice?
Mbe-var-osac(3):
(processing a bucket)

- **Input:**
  
  \[ B_i = \{ f \in F \mid X_i \in \text{scope}(f) \} = \{ f_{i1}, \ldots, f_{it} \} \]
  
  \[ p^f_{x_i}, \forall f, \forall x_i \in D_i \quad c_i(x_i), u_i \]

- **Solve Ipvar:**
  Max \( u_i \) \( \forall x_i \in D_i, c_i(x_i) - u_i + \sum_{f \in B_i} p^f_i(x_i) \geq 0 \)
  (shifts only to X_i)

  \[ \forall f \in B_i, \forall x \in D_f, f(x) - p^f_i(x[i]) \geq 0 \]

- **Output:**
  \[ B_i' = \{ c_i(x_i), f_{i1}', \ldots, f_{ir}' \} s.t. f_j'(x) = f_j(x) - p_{x[i]} \]

- Apply mini-bucket to the output of LPVAR in bucket i
- Properties: processing LP-var seems easy.
- Are we guaranteed to improve?
Linear relaxation-based schemes (MPLP class, Globerson and Jakkola)

Find: \( \mathbf{x} = (x_1, \ldots, x_n) \) to all the variables which maximizes the sum of the factors:

\[
\text{MAP}(\theta) = \max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(x_f). \tag{1.1}
\]

Best upper bound by Equivalence preserving transformations:

\[
\min_{\delta} L(\delta), \tag{1.2}
\]

\[
L(\delta) = \sum_{i \in V} \max_{x_i} \left( \theta_i(x_i) + \sum_{f : i \in f} \delta_{fi}(x_i) \right) + \sum_{f \in F} \max_{x_f} \left( \theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i) \right).
\]

\( \delta_{fi}(x_i) \) is the cost shifted from \( f \) to value \( x_i \) of \( X_i \).

There are several variations of scheme computing the optimizing shifts based on partial gradient descent, which differ by what is being kept constant. The 1.2 task is the Dual of a linear relaxation of the original problem.
MPLP block coordinated descent

Inputs:
- A set of factors $\theta_i(x_i), \theta_f(x_f)$.

Output:
- An assignment $x_1, \ldots, x_n$ that approximates the MAP.

Algorithm:
- Initialize $\delta_{fi}(x_i) = 0$, $\forall f \in F, i \in f, x_i$.
- Iterate until small enough change in $L(\delta)$ (see Eq. 1.2):
  - For each $f \in F$, perform the updates
    \[
    \delta_{fi}(x_i) = -\delta_i^{-f}(x_i) + \frac{1}{|f|} \max_{x_f \setminus i} \left[ \theta_f(x_f) + \sum_{i \in f} \delta_i^{-f}(x_i) \right],
    \]  
    simultaneously for all $i \in f$ and $x_i$. We define $\delta_i^{-f}(x_i) = \theta_i(x_i) + \sum_{f \neq f} \delta_{fi}(x_i)$.
- Return $x_i \in \arg \max_{\hat{x}_i} \hat{\theta}_i^\delta(\hat{x}_i)$ (see Eq. 1.6).

Return updated functions:
- $\tilde{\theta}_i^\delta(x_i) = \theta_i(x_i) + \sum_{f: i \in f} \delta_{fi}(x_i)$
- $\tilde{\theta}_f^\delta(x_f) = \theta_f(x_f) - \sum_{i \in f} \delta_{fi}(x_i)$.
\[
\delta_{f_i}(x_i) = -\delta_{i}^{-f}(x_i) + \frac{1}{|f|} \max_{x_{f \setminus i}} \left[ \theta_f(x_f) + \sum_{i \in f} \delta_i^{-f}(x_i) \right],
\]
EMPLP and NMPLP

Inputs: A graph $G = (V, E)$, potential functions $\theta_{ij}(x_i, x_j)$ for each edge $ij \in E$.

Initialization: Initialize messages to any value.

Algorithm:
- Iterate until a stopping criterion is satisfied:
  - Max-product: Iterate over messages and update ($c_{ji}$ shifts the max to zero)
    $$m_{ji}(x_i) \leftarrow \max_{x_j} \left[ m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] - c_{ji}$$
  - EMPLP: For each $ij \in E$, update $\lambda_{ji}(x_i)$ and $\lambda_{ij}(x_j)$ simultaneously (the update for $\lambda_{ij}(x_j)$ is the same with $i$ and $j$ exchanged)
    $$\lambda_{ji}(x_i) \leftarrow -\frac{1}{2} \lambda_i^{-j}(x_i) + \frac{1}{2} \max_{x_j} \left[ \lambda_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right]$$
  - NMPLP: Iterate over nodes $i \in V$ and update all $\gamma_{ij}(x_j)$ where $j \in N(i)$
    $$\gamma_{ij}(x_j) \leftarrow \max_{x_i} \left[ \theta_{ij}(x_i, x_j) - \gamma_{ji}(x_i) + \frac{2}{|N(i)| + 1} \sum_{k \in N(i)} \gamma_{ki}(x_i) \right]$$
- Calculate node “beliefs”: Set $b_i(x_i)$ to be the sum of incoming messages into node $i \in V$ (e.g., for NMPLP set $b_i(x_i) = \sum_{k \in N(i)} \gamma_{ki}(x_i)$).

Output: Return assignment $x$ defined as $x_i = \arg \max_{x_i} b(\hat{x}_i)$.

Figure 1: The max-product, EMPLP and NMPLP algorithms. Max-product, EMPLP and NMPLP use messages $m_{ij}, \lambda_{ij}$ and $\gamma_{ij}$ respectively. We use the notation $m_{j}^{-i}(x_j) = \sum_{k \in N(j) \setminus i} m_{kj}(x_j)$. 
Convergence Properties

Proposition 3 If the fixed point of MPLP has $b_i(x_i)$ such that for all $i$ the function $b_i(x_i)$ has $a_i$, then $x$ is the solution to the MAP problem and the LP relaxation is exact.

Proposition 4 When $x_i$ are binary, the MPLP fixed point can be used to obtain the primal optimum.
Generalized MPLP

\[ \lambda_{c \rightarrow s}(x_s) = -\left(1 - \frac{1}{|\mathcal{S}(c)|}\right)\lambda_s^{-c}(x_s) + \frac{1}{|\mathcal{S}(c)|}\max_{x_{c \setminus s}} \left[ \sum_{\hat{s} \in \mathcal{S}(c) \setminus s} \lambda_{\hat{s}}^{-c}(x_{\hat{s}}) + \theta_c(x_c) \right] \]
MBE+MPLP

• MPE-MPLP can be combined in a similar ways as with OSAC. Which method can provide guarantee for improvement....

• Use generalized MPLP as a basis...

• Develop directly similar to Alex and Qiang for belief.
Generalized MPLP

\[
\lambda_{c \to s}(x_s) = -\left(1 - \frac{1}{|S(c)|}\right)\lambda^c_s(x_s) + \frac{1}{|S(c)|} \max_{x_{c\setminus s}} \left[ \sum_{\tilde{s} \in S(c) \setminus s} \lambda^c_{\tilde{s}}(x_{\tilde{s}}) + \theta_c(x_c) \right]
\]

Perhaps re-derive generalized MPLP where S (separators) are just single-variables. We can do that by using a join-graph generated using the mini-bucket schematic method. In this case the separators within each mini-bucket are the bucket-variable itself.

Considering a covering graph that corresponds to the mini-bucket structure, we can have a collection of shifting costs from mini-bucket to single variables and we can Derive the updating equations associated with the dual that corresponds to this problem. The execution of this algorithm in an ordered derived from the mini-bucket may be a mini-bucket-like algorithm that iterates. (This may be actually identical to Alex and Qiang)
Who wants to grab this?

THE END
Local Consistency in Constraint Networks

• Massive local inference

  – Time efficient (local inference, as mini buckets)
  – Infer only small constraints, added to the network
  – No variable is eliminated
  – Produces an equivalent more explicit problem
  – May detect inconsistency (prune tree search)

Arc consistency inference in the scope of 1 constraint
Arc Consistency (binary CSP)

- for a constraint $c_{AB}$ and variable $A$

- Applied iteratively on all constraint/variables
- Confluent, incremental, complexity in $O(md^2)$
- Empty domain $\Rightarrow$ inconsistency
Arc Consistency and Cost Functions

• for a cost function $f_{AB}$ and a variable $A$

EQUIVALENCE LOST
Shifting Costs (cost compensation)

Subtract from source in order to preserve the problem

⇒ Equivalence Preserving Transformation
Complete Inference vs Local Inference

- Complete inference
  - Local consistency

- Combine, eliminate, add & forget
- Systematic inference
- Exponential time/space
- Preserves optimum

- Provides the optimum $f_\emptyset$

- Combine, eliminate, add & subtract
- Massive local inference
- Space/time efficient
- Preserves equivalence

- Provides a lb $f_\emptyset$
Equivalence Preserving Transformation

• Shifting costs from $f_{AB}$ to A
  \[ \text{Shift}(f_{AB}, (A, w), 1) \]

\begin{center}
\begin{tikzpicture}[node distance=2cm,>=latex]
  \node (A) at (0,0) [circle,draw] {A};
  \node (B) at (2,0) [circle,draw] {B};
  \node (v) at (1,1) [circle,draw] {v};
  \node (w) at (1,-1) [circle,draw] {w};
  \draw[->] (A) to node [above] {1} (v);
  \draw[->] (A) to node [below] {1} (w);
  \draw[->] (v) to node [right] {1} (B);
  \draw[->] (w) to node [right] {0} (B);
\end{tikzpicture}
\end{center}

**Arc EPT:** shift cost in the scope of 1 cost function

Problem structure preserved

• Can be reversed (e.g. \[ \text{Shift}(f_{AB}, (A, w), -1) \])
Equivalence Preserving Transformations

- EPTs may cycle
- EPTs may lead to different $f_0$

- Which EPTs should we apply?

$f_{\emptyset} = 1$
Local Consistency

– Equivalence Preserving Transformation
– Chaotic iteration of EPTs
– Optimal set of EPTs
– Improving sequence of EPTs
Local Consistency

– Equivalence Preserving Transformation

– Chaotic iteration of EPTs
  • Enforce a local property by one or two EPT(s)

– Optimal set of EPTs

– Improving sequence of EPTs
Node Consistency (NC*)

- For any variable A
  
  - $\forall a, f_{\emptyset} + f_A(a) < k$
  
  - $\exists a, f_A(a) = 0$

- Complexity:
  $O(nd)$

\[ \text{Shift}(f_C, \emptyset, 1) \]
\[ \text{Shift}(f_A, \emptyset, -1); \text{Shift}(f_A, \emptyset, 1) \]

(Larrosa, AAAI 2002)
Arc Consistency (AC*)

- NC*
- For any $f_{AB}$
  - $\forall a \exists b$
  
  $$f_{AB}(a,b) = 0$$

- $b$ is a support
- complexity:
  $$O(n^2d^3)$$

**Example:**

Switching:
- $f_{AC}, (C,v), 2$
- $f_{BC}, (B,v), 1$
- $f_{CB}, \emptyset, 1$
- $f_{BC}, (B,w), 1$

(Schiex, CP 2000)

(Larrosa, AAAI 2002)
Directional AC (DAC\*)

- NC\*
- For all $f_{AB}$ ($A<B$)
  - $\forall a \ni b$
  - $f_{AB}(a,b) + f_B(b) = 0$
- $b$ is a full-support
- complexity:
  - $O(ed^2)$

Shift($f_{BC},(C,v),-1$)  Shift($f_{BC},(B,v),1$)
Shift($f_A,\emptyset,-2$)  Shift($f_A,\emptyset,2$)

(Cooper, Fuzzy Sets and Systems 2003)
DAC lb = Mini-Bucket(2) lb

A < E < D < C < B

- DAC provides an equivalent problem: incrementality
- DAC+NC (value pruning) can improve lb
Other « Chaotic » Local Consistencies

- **FDAC* = DAC+AC+NC**
  - Stronger lower bound
  - $O(end^3)$
  - Better compromise

- **EDAC* = FDAC+ EAC (existential AC)**
  - Even stronger
  - $O(ed^2 \cdot \max\{nd, k\})$
  - Currently among the best practical choice

References:
(Cooper, Fuzzy Sets and Systems 2003)
(Larrosa & Schiex, IJCAI 2003)
(Larrosa & Schiex, AI 2004)
(Cooper & Schiex, AI 2004)
(Heras et al., IJCAI 2005)
(Sanchez et al, Constraints 2008)
Local Consistency

- Equivalence Preserving Transformation
- Chaotic iteration of EPTs
- Optimal set of simultaneously applied EPTs
  - Solve a linear problem in rational costs
- Improving sequence of EPTs
Finding an EPT Sequence Maximizing the LB

Bad news

Finding a sequence of integer arc EPTs that maximizes the lower bound defines an NP-hard problem

(Cooper & Schiex, AI 2004)
Good news: A continuous linear formulation

- $u_A$: cost shifted from $A$ to $f_0$
- $p_{Aa}^{AB}$: cost shifted from $f_{AB}$ to $(A,a)$

$$\max \sum u_i$$

Subject to non negativity of costs

$$\forall i \in X, \forall a \in d_i, \quad c_i(a) - u_i + \sum_{(c_S \in C'), (i \in S)} p_{i,a}^S \geq 0$$

$$\forall c_S \in C, |S| > 1, \forall t \in \ell(S) \quad c_S(t) - \sum_{i \in S} p_{i,t[\{i\}]}^S \geq 0$$

$n + m.r.d$ variables
$n.d + m.d'$ linear constraints
Optimal Soft AC

solved by Linear Programming

• Polynomial time, rational costs (bounded arity $r$)

• Computes an optimal set of EPT $(u_A, p_{Aa}^{AB})$ to apply simultaneously

• Stronger than AC, DAC, FDAC, EDAC...
  (or any local consistency that preserves scopes)

(Cooper et al., IJCAI 2007)
(Schlesinger, Kibernetika 1976)
(Boros & Hammer, Discrete Appl. Math. 2002)
\[ f_0 = 1 \]

AC, DAC, FDAC, EDAC

\[ p_{2c}^{23} = p_{3a}^{34} = p_{3b}^{31} = p_{1a}^{12} = p_{1c}^{14} = -1 \]

\[ p_{3a}^{23} = p_{3b}^{23} = p_{4c}^{34} = p_{1a}^{31} = p_{1c}^{31} = p_{2c}^{12} = p_{4a}^{14} = u_4 = 1 \]
Local Consistency

- Equivalence Preserving Transformation
- Chaotic iteration of EPTs
- Optimal set of EPTs
- Improving sequence of EPTs
  - Find an improving sequence using classical arc-consistency in classical CSPs