Extensions of submodularity and their application in computer vision

Vladimir Kolmogorov

IST Austria

Oxford, 20 January 2014
Linear Programming relaxation

\[ f(x) = \sum_{i} f_i(x_i) + \sum_{A} f_A(x_A) \quad \forall x_i \in D \]

- Popular approach: *Basic LP relaxation* (BLP)
  - for pairwise energies: *Schlesinger’s LP*
  - for higher-order energies: enforce consistency between variables \( \mu_A \) and \( \mu_i \) for all \( A \) and \( i \in A \)
  - many algorithms for (approximately) solving it (MSD, TRW-S, MPLP, …)
Tightness of BLP

\[ f(x) = \sum_{i} f_i(x_i) + \sum_{A} f_A(x_A) \]

**Theorem** [Cooper’08], [Werner’10]. If each term \( f_A \) is a submodular function then BLP relaxation is tight.

- Other such classes? Complete classification?
- Use VCSP framework
Valued Constraint Satisfaction Problem (VCSP)

- Language $\Gamma$: a set of cost functions $f : D^m \rightarrow \mathbb{Q}_+ \cup \{\infty\}$

- VCSP($\Gamma$): class of functions that can be expressed as a sum of functions from $\Gamma$ with overlapping sets of vars
  - Goal: minimize this sum

- Complexity of $\Gamma$?
- Does BLP solves $\Gamma$?

- Feder-Vardi conjecture (for CSPs):
  
  Every CSP language is either tractable or NP-hard

  - CSP: contains functions $f : D^m \rightarrow \{0, \infty\}$
Classifications for finite-valued CSPs

**Theorem** [Thapper, Živný FOCS’12], [K ICALP’13]

BLP solves $\Gamma$ iff it admits a *binary symmetric fractional polymorphism*

- Other languages are NP-hard [Thapper, Živný STOC’13]
Submodular functions

\[ a \sqcap b = \min\{a, b\} \]

\[ a \sqcup b = \max\{a, b\} \]

\[ f(x \sqcap y) + f(x \sqcup y) \leq f(x) + f(y) \]
New classes of functions

\[ \sum \omega(\sqcap) f(x \sqcap y) \leq \frac{1}{2} f(x) + \frac{1}{2} f(y) \]

\( \omega: \) distribution over symmetric operations

\( \sqcap : D \times D \rightarrow D \)
Useful classes

• Submodular functions
  - Pairwise functions: can be solved via maxflow ("graph cuts")
  - Lots of applications in computer vision

• Bisubmodular functions
  - Obtaining partial optimal solutions
  - Characterize extensions of "QPBO" to arbitrary pseudo-Boolean functions [K’10,12]

• $k$-submodular functions
  - Partial optimality for functions of $k$-valued variables
  - This talk: efficient algorithm for Potts energy [Gridchyn, K ICCV’13]
Partial optimality

- Input: function $f : D^n \rightarrow \mathbb{R}$
- Partial labeling $x$ is *optimal* if it can be extended to a full optimal labeling $x^* \in \text{arg min } f$
Partial optimality

- Input: function $f : D^n \rightarrow \mathbb{R}$

- Partial labeling $x$ is *optimal* if it can be extended to a full optimal labeling $x^* \in \text{arg min } f$

- Can be viewed as a labeling $x \in \hat{D}^n$, $\hat{D} = D \cup \{\perp\}$
$k$-submodular relaxations

- Input: function $f : D^n \rightarrow \mathbb{R}$

$D = \{1, \ldots, k\}$
**$k$-submodular relaxations**

- **Input**: function $f : D^n \rightarrow \mathbb{R}$
  $$D = \{1, \ldots, k\}$$

- Construct extension $g : \hat{D}^n \rightarrow \mathbb{R}$, $\hat{D} = D \cup \{\perp\}$
  which is $k$-submodular

- Minimize $g$

---

**Theorem:**
Minimum of $g$ partially optimal
$k$-submodularity

• Function $g : D^n \rightarrow \mathbb{R}$ is $k$-submodular if

$$g(x \cap y) + g(x \cup y) \leq g(x) + g(y)$$

$$(a \sqcap b, a \sqcup b) = \begin{cases} (\bot, \bot) & \text{if } a \neq b \text{ and } a, b \neq \bot \\ (\min\{a, b\}, \max\{a, b\}) & \text{otherwise} \end{cases}$$
$k$-submodular relaxations

- **Case** $k = 2$ ([K’10,12])
  - **Bisubmodular relaxation**
  - Characterizes extensions of QPBO

- **Case** $k > 2$
  - [Gridchyn, K ICCV’13] :
    - efficient method for Potts energies
  - [Wahlström SODA’14] :
    - used for FPT algorithms
$k$-submodular relaxations for Potts energy

\[ f(x) = \sum_{i} f_i(x_i) + \sum_{i,j} \lambda_{ij} [x_i \neq x_j] \]

- $k$-submodular relaxation:

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j) \]
$k$-submodular relaxations for Potts energy

$$f(x) = \sum_{i} f_i(x_i) + \sum_{i,j} \lambda_{ij} [x_i \neq x_j]$$

- $k$-submodular relaxation:

$$g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j)$$

$d(a, b) : \text{tree metric}$
$k$-submodular relaxations for Potts energy

\[ f(x) = \sum_i f_i(x_i) + \sum_{i,j} \lambda_{ij} [x_i \neq x_j] \]

- $k$-submodular relaxation:

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j) \]

$g_i(\cdot) : k$-submodular relaxation of $f_i(\cdot)$
$k$-submodular relaxations for Potts energy

$$g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j)$$

- Minimizing $g : O(\log k)$ maxflows
- Alternative approach: [Kovtun ’03,’04]
  - Stronger than $k$-submodular relaxations (labels more)
  - Can be solved by the same approach!
    ➢ complexity: $k \Rightarrow O(\log k)$ maxflows
- Part of “Reduce, Reuse, Recycle” [Alahari et al.’08,’10]
- Our tests for stereo: 50-93% labeled
  ➢ with 9x9 windows
- Speeds up alpha-expansion for unlabeled part
Tree Metrics

- [Felzenszwalb et al.'10]: \( O(\log k) \) maxflows for

\[
g(x) = \sum_i \lambda_i d(x_i, c_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j)
\]
Tree Metrics

- [Felzenszwalb et al.'10]: $O(\log k)$ maxflows for
  \[
g(x) = \sum_i \lambda_i d(x_i, c_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j)
  \]

- [This work]: extension to more general unary terms
  - new proof of correctness
Special case: Total Variation

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

- Convex unary terms
- Reduction to parametric maxflow \[\text{[Hochbaum’01]}, \text{[Chambolle’05]}, \text{[Darbon, Sigelle’05]}\]
New condition: \( T \)-convexity

\[
g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j)
\]

- Convexity for any pair of adjacent edges:
Algorithm: divide-and-conquer

- Pick edge \((a, b)\)
- Compute \(\text{arg min}\{g(x) \mid x \in \{a, b\}^n\}\)
Algorithm: divide-and-conquer

- Pick edge \((a, b)\)
- Compute \(\arg \min \{g(x) \mid x \in \{a, b\}^n\}\)
- Claim: \(g\) has a minimizer as shown below
- Solve two subproblems recursively
Achieving balanced splits

- For star graphs, all splits are unbalanced

- Solution [Felzenszwalb et al.’10]: insert a new short edge
  - modify unary terms $g_i(\cdot)$ accordingly
Algorithm illustration

- $k = 7$ labels:

1, 2, 3, 4, 5, 6, 7
Algorithm illustration

- $k = 7$ labels:
Algorithm illustration

- $k = 7$ labels:
Algorithm illustration

- $k = 7$ labels:
Algorithm illustration

- $k = 7$ labels:

- $\lceil \log_2 k \rceil + 1$ maxflows

- “Kovtun labeling”

- unlabeled part, run alpha-expansion
Stereo results

Kovtun's labeling:

alpha expansion:

ground truth:
Proof of correctness (sketch)

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]
Proof of correctness (sketch)

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

- For labeling \( x \) and edge \((a, b)\) define \( x^{[ab]} \in \{a, b\}^n \)
Proof of correctness (sketch)

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

• For labeling \( x \) and edge \( (a, b) \) define \( x^{[ab]} \in \{a, b\}^n \)
Proof of correctness (sketch)

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

- Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + \text{const} \]
Proof of correctness (sketch)

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

• Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + \text{const} \]
Proof of correctness (sketch)

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

- Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + \text{const} \]
Proof of correctness (sketch)

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij}|x_j - x_i| \]

• Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + const \]
Proof of correctness (sketch)

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

• Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + const \]
Proof of correctness (sketch)

\[ g(x) = \sum_i g_i(x_i) + \sum_{i,j} \lambda_{ij} |x_j - x_i| \]

• Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + \text{const} \]

• Equivalent problem: minimize \( \sum_{(a,b)} g(y^{ab}) \)

  with \( y^{ab} \in \{a, b\}^n \) subject to consistency constraints

• Equivalent to independent minimizations of \( g(y^{ab}) \)

  - consistency holds automatically due to convexity of \( g_i(\cdot) \)
Extension to trees

\[ g(x) = \sum_{i} g_i(x_i) + \sum_{i,j} \lambda_{ij} d(x_i, x_j) \]

- Coarea formula:

\[ g(x) = \sum_{(a,b)} g(x^{[ab]}) + \text{const} \]
Summary

Part I:
• New tractable class of functions
  ➢ complete characterization for finite-valued CSPs

Part II:
• $k$-submodular relaxations for partial optimality
• For Potts model:
  ➢ cast Kovtun’s approach as $k$-submodular function minimization
  ➢ $O(\log k)$ algorithm
  ➢ generalized alg. of [Felzenswalb et al’10] for tree metrics

• Future work: $k$-submodular relaxations for other functions?
postdoc & PhD student positions are available