

Partial Optimality via Iterative Pruning for the Potts Model

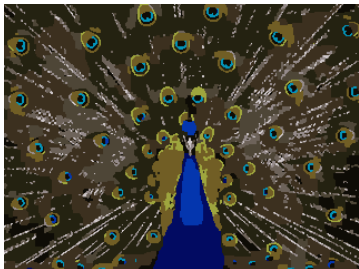
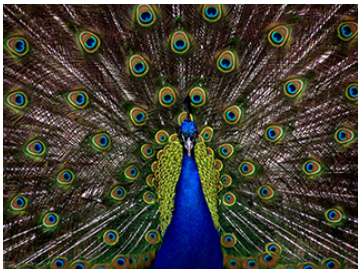
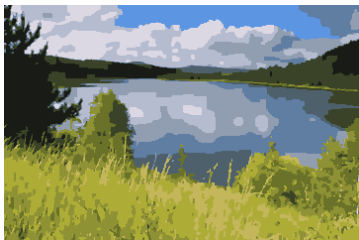
Paul Swoboda, Bogdan Savchynskyy, Jörg Hendrik Kappes and Christoph Schnörr

*Image & Pattern Analysis Group
University of Heidelberg*

June 3, 2013

Fourth International Conference on Scale Space and Variational Methods
in Computer Vision

Applications of energy minimization problems: segmentation



and many others: optical flow, stereo, . . .

Continuous energy:

$$\min_{u \in BV(\Omega; \{1, \dots, k\})} E_{cont} = \int_{\Omega} |Du| + W(x, u(x)) dx .$$

Continuous energy:

$$\min_{u \in BV(\Omega; \{1, \dots, k\})} E_{cont} = \int_{\Omega} |Du| + W(x, u(x)) dx .$$

Discrete Potts energy:

$$\min_{u_a \in \{e_1, \dots, e_k\} \forall a \in V} E(u) = \sum_{a \in V} \sum_{l=1}^k \theta_a(l) u_a(l) + \sum_{(a,b) \in E} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)| .$$

where $G = (V, E)$ is a graph.

NP-hard

Tractable relaxation

Continuous energy:

$$\min_{u \in BV(\Omega; \Delta_k)} E_{cont} = \int_{\Omega} |Du| + W(x, u(x)) dx .$$

Discrete Potts energy:

$$\min_{u_a \in \Delta_k \forall a \in V} E(u) = \sum_{a \in V} \sum_{l=1}^k \theta_a(l) u_a(l) + \sum_{(a,b) \in E} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)| .$$

where $G = (V, E)$ is a graph.

Polynomial time solvable

Solution of relaxation at red points not integral anymore:



Solution of relaxation at red points not integral anymore:

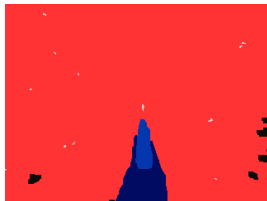


Partial Optimality:

Let $u^* \in \operatorname{argmin}_{u \in \{e_1, \dots, e_k\}} E(u)$

$u_{relax}^* \in \operatorname{argmin}_{u \in \Delta_k^{|V|}} E(u)$

Does $u_{relax}^*(a) \in \{e_1, \dots, e_k\} \Rightarrow u_{relax}^*(a) = u^*(a)$ hold?



Benefits of partial optimality:

- Obtain *integral* solution by solving the remaining variables with exact methods¹.
- Speed up minimization²: Run an algorithm, stop after a few iterations. Check for partial optimality. Iterate.

¹Kappes et al., “Towards Efficient and Exact MAP-Inference for Large Scale Discrete Computer Vision Problems via Combinatorial Optimization”.

²Alahari, Kohli, and Torr, “Reduce, Reuse & Recycle: Efficiently Solving Multi-Label MRFs”.

Related work concerning partial optimality:

- Nemhauser and Trotter, “Vertex packings: Structural properties and algorithms”
- Boros and Hammer, “Pseudo-Boolean optimization”
- Rother et al., “Optimizing Binary MRFs via Extended Roof Duality”
- Kohli et al., “On partial optimality in multi-label MRFs”
- Windheuser, Ishikawa, and Cremers, “Generalized Roof Duality for Multi-Label Optimization: Optimal Lower Bounds and Persistency”
- Kahl and Strandmark, “Generalized roof duality”
- Kovtun, “Partial Optimal Labeling Search for a NP-Hard Subclass of $(\max,+)$ Problems”

Partial Optimality criterion: Given $A \subset V$, a labeling $u^*_{|A}$ on A is partially optimal if for every labeling $u^{outside}$ on $V \setminus A$ it holds that $u^*_{|A} \in \operatorname{argmin}_{\{u: u_{|V \setminus A} = u^{outside}\}} E(u)$.

Partial Optimality criterion: Given $A \subset V$, a labeling $u^*_{|A}$ on A is partially optimal if for every labeling $u^{outside}$ on $V \setminus A$ it holds that $u^*_{|A} \in \operatorname{argmin}_{\{u: u_{|V \setminus A} = u^{outside}\}} E(u)$.

Tractable Partial Optimality Criterion: Bound away the effect of all labelings on $V \setminus A$ and test.

Partial Optimality criterion: Given $A \subset V$, a labeling $u|_A^*$ on A is partially optimal if for every labeling $u^{outside}$ on $V \setminus A$ it holds that $u|_A^* \in \operatorname{argmin}_{\{u: u|_{V \setminus A} = u^{outside}\}} E(u)$.

Tractable Partial Optimality Criterion: Bound away the effect of all labelings on $V \setminus A$ and test.

Algorithmic idea: Prune nodes of the graph G until we arrive at a set which has a labeling fulfilling the tractable partial optimality criterion.

Original energy:

$$E(u) = \sum_{a \in V} \sum_{l=1}^k \theta_a(l) u_a(l) + \sum_{(a,b) \in E} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)|.$$

Modified energy for a subset A and labeling \tilde{u} :

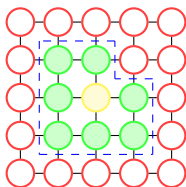
$$E_{A, \tilde{u}}(u) = \sum_{a \in A} \sum_{l=1}^k \tilde{\theta}_a(l) u_a(l) + \sum_{(a,b) \in E, a,b \in A} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)|.$$

Modified energy for a subset A and labeling \tilde{u} :

$$E_{A,\tilde{u}}(u) = \sum_{a \in A} \sum_{l=1}^k \tilde{\theta}_a(l) u_a(l) + \sum_{(a,b) \in E, a,b \in A} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)|.$$

For every edge $(a, b) \in E$ with $a \in A$, $b \notin A$ modify the unary costs

$$\tilde{\theta}_a = \begin{cases} \theta_a(i) + \alpha_{ab} & , \tilde{u}_a(i) = 1 \\ \theta_a(i) & , \tilde{u}_a(i) = 0 \end{cases} .$$



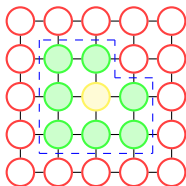
Intuition: We worsen the unaries for the current labeling.

Modified energy for a subset A and labeling \tilde{u} :

$$E_{A, \tilde{u}}(u) = \sum_{a \in A} \sum_{l=1}^k \tilde{\theta}_a(l) u_a(l) + \sum_{(a,b) \in E, a, b \in A} \sum_{l=1}^k \frac{\alpha_{ab}}{2} |u_a(l) - u_b(l)|.$$

For every edge $(a, b) \in E$ with $a \in A$, $b \notin A$ modify the unary costs

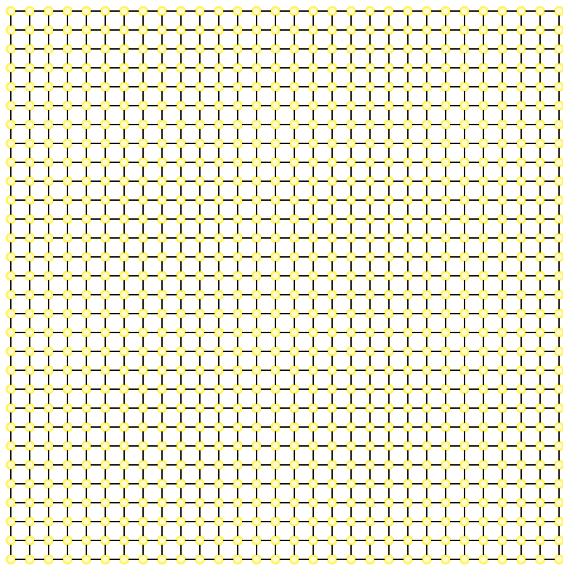
$$\tilde{\theta}_a = \begin{cases} \theta_a(i) + \alpha_{ab} & , \tilde{u}_a(i) = 1 \\ \theta_a(i) & , \tilde{u}_a(i) = 0 \end{cases}.$$



Intuition: We worsen the unaries for the current labeling.

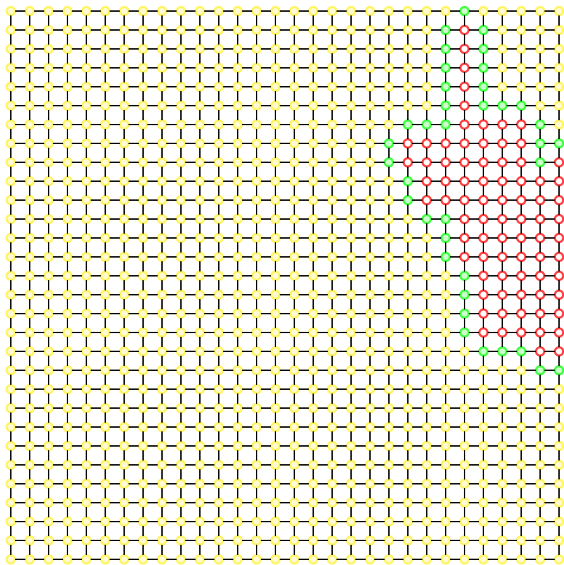
Theorem: If u is optimal for the problem with modified unaries, then it is partially optimal.

Iteration 0



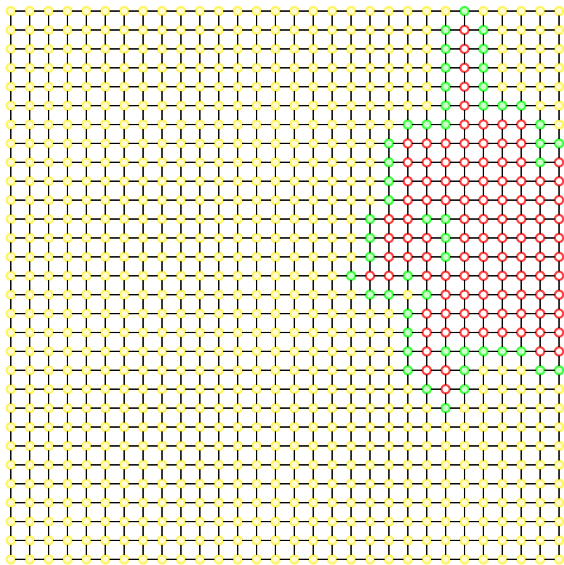
- Outside node
- Inside node
- Boundary node

Iteration 1



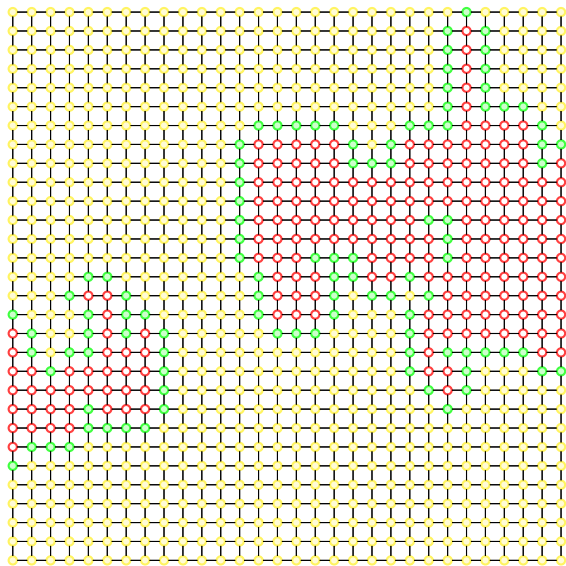
- Outside node
- Inside node
- Boundary node

Iteration 2



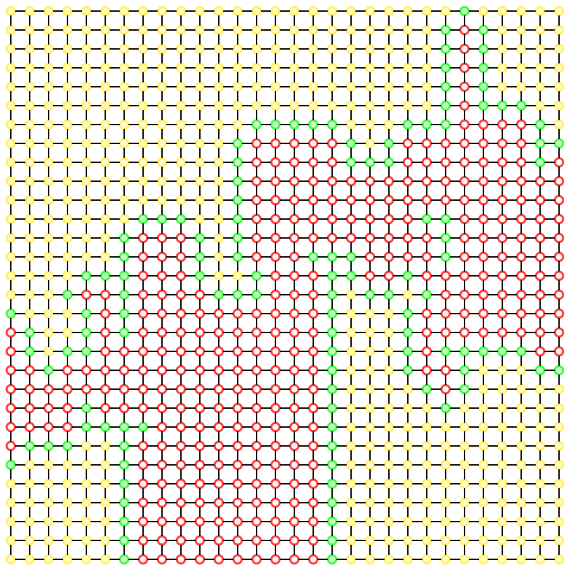
- Outside node
- Inside node
- Boundary node

Iteration 3



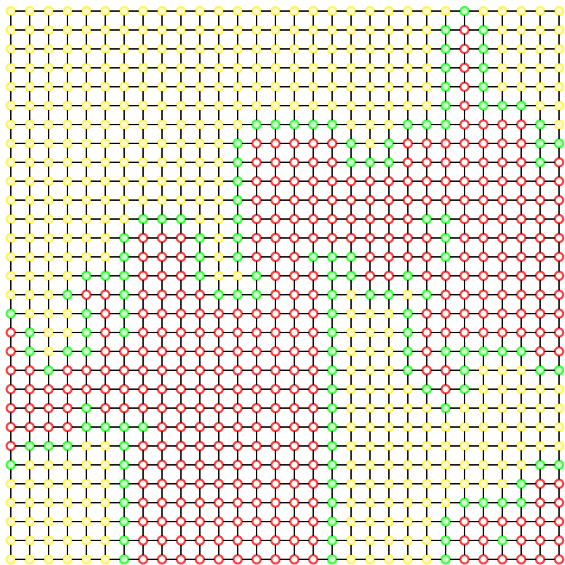
- Outside node
- Inside node
- Boundary node

Iteration 4



- Outside node
- Inside node
- Boundary node

Iteration 5



- Outside node
- Inside node
- Boundary node

Algorithm 1: Finding persistent variables

Compute solution of the relaxed problem on V .

Prune all non-integral variables.

while *Variables had to be pruned* **do**

 Modify unary costs.

 Compute solution of the relaxed problem on current set.

 Prune all non-integral variables.

 Prune all variables that have changed since last iteration.

end

We compared our approach with the following methods:

- MQPBO³.
- Kovtun's method⁴.
- KMPQBO: Apply Kovtun's method followed by MQPBO.
- KMPQBO- N : Apply Kovtun's method followed by N iterations of MQPBO.

We used the OpenGM⁵ software package for these implementations. All models were taken from the OpenGM benchmark website⁶.

³Kohli et al., "On partial optimality in multi-label MRFs".

⁴Kovtun, "Partial Optimal Labeling Search for a NP-Hard Subclass of (max,+) Problems".

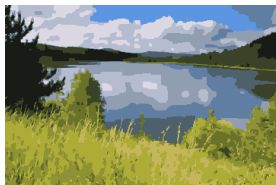
⁵*OpenGM*. hci.iwr.uni-heidelberg.de/opengm2/.

⁶*OpenGM benchmark*.

<http://hci.iwr.uni-heidelberg.de/opengm2/?l0=benchmark>.

Color segmentation dataset⁷:

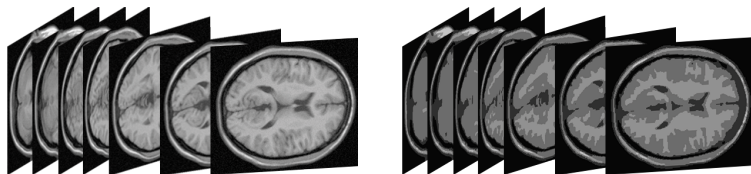
Dataset	Ours	KMQPBO	KMQPBO100	Kovtun	MQPBO
clownfish (12)	0.9852	0.7659	0.9495	0.7411	0.0467
crops (12)	0.9308	0.6486	0.8803	0.6470	0.0071
fourcolors(4)	0.9993	0.6952	0.7010	0.6952	0.0
lake (12)	0.9998	0.7613	0.9362	0.7487	0.0665
palm (12)	0.8514	0.6866	0.7192	0.6865	0.0
penguin (8)	0.9999	0.9240	0.9471	0.9199	0.0103
peacock (12)	0.1035	0.0559	0.1234	0.0559	0.0
snail (3)	0.9997	0.9786	0.9819	0.9778	0.5835
strawberry-glass (12)	0.9639	0.5502	0.5997	0.5499	0.0



⁷Lellmann and Schnörr, “Continuous Multiclass Labeling Approaches and Algorithms” .

Brain scan dataset⁸:

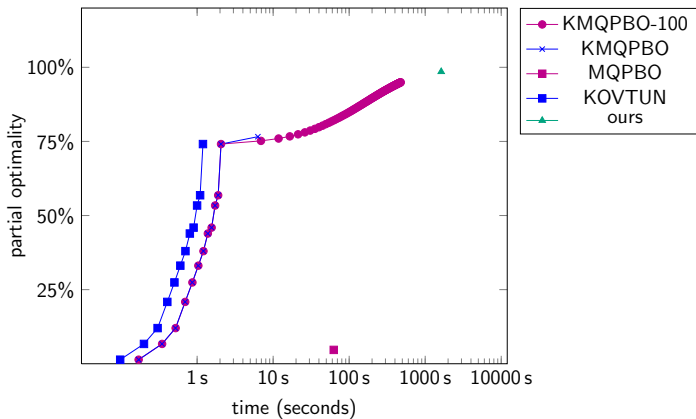
Dataset	Ours	KMQPBO	KMQPBO100	Kovtun	MQPBO
$181 \times 217 \times 20$	0.9968	0.9993	0.9994	0.9235	0.3886
$181 \times 217 \times 26$	0.9969	1	0.9996	0.9322	0.3992
$181 \times 217 \times 36$	0.9967	†	†	0.9363	0.4020
$181 \times 217 \times 60$	0.9952	†	†	0.9496	0.4106



⁸*BrainWeb: Simulated Brain Database.*

<http://brainweb.bic.mni.mcgill.ca/brainweb/>.

Partial optimality over time:



Conclusion

- Extend our approach to more general labeling problems.

Conclusion

- Extend our approach to more general labeling problems.
- Improve computational efficiency

Conclusion

- Extend our approach to more general labeling problems.
- Improve computational efficiency
- Layered approach:
 1. Kovtun's method
 2. Our method
 3. ILP solver