Optimization

Computer Vision, Lecture 13

Per-Erik Forssén (slides by Michael Felsberg)
Computer Vision Laboratory
Department of Electrical Engineering

LINKÖPINGS UNIVERSITET

Optimization: Overview

Function		**Output (codomain / target set)**	
	Set	Continuous	Discrete
Input (domain of definition)	Continuous	Lecture 15	Lecture 15
	Discrete	Lecture 13	Lecture 13

e.g.: stereo e.g.: segmentation

Why Optimization?

- Computer vision algorithms are usually very complex
  - Many parameters (dependent)
  - Data dependencies (non-linear)
  - Outliers and occlusions (noise)
- Classical approach
  - Trial and error (hackers' approach)
  - Encyclopedic knowledge (recipes)
  - Black-boxes + glue (hide problems)

Why Optimization?

- Establishing CV as scientific discipline
  - Derive algorithms from first principles (*optimal solution*)
  - Automatic choice of parameters (*parameter free*)
  - Systematic evaluation (*benchmarks on standard datasets*)

Optimization: howto

1. Choose a *scalar* measure (objective function) of success
   - From the benchmark
   - Such that optimization becomes *feasible*
   - Project functionality onto *one dimension*
2. Approximate the world with a model
   - Definition: allows to make *predictions*
   - Purpose: makes optimization *feasible*
   - Enables: *proper* choice of dataset

Similar to economics (money rules)

Optimization: howto

3. Apply suitable framework for model fitting
   - This lecture
   - Systematic part (1 & 2 are ad hoc)
   - Current focus of research
4. Analyze resulting algorithm
   - Find *appropriate* dataset
   - Ignore runtime behavior (*highly non-optimized Matlab code*) ;-)

Examples

- Relative pose (F-matrix) estimation:
  - Algebraic error (quadratic form)
  - Linear solution by SVD
  - Robustness by random sampling (RANSAC)
  - Result: F and inlier set
- Bundle adjustment
  - Geometric (reprojection) error (quadratic error)
  - Iterative solution using LM
  - Result: camera pose and 3D points

Taxonomy

- Objective function
  - Domain/manifold (algebraic error, geometric error, data dependent)
  - Robustness (explicitly in error norm, implicitly by Monte-Carlo approach)
- Model / simplification
  - Linearity (limited order), Markov property, regularization
- Algorithm
  - Approximate / analytic solutions (minimal problem)
  - Minimal solutions (over-determined)

Taxonomy example: KLT

- Objective function
  - Domain/manifold: grey values / RGB / …
  - Robustness: no (quadratic error, no regularization)

$$\varepsilon(\mathbf{d}) = \sum_{\mathbf{x} \in \mathcal{N}} w(\mathbf{x}) |f(\mathbf{x} - \mathbf{d}) - g(\mathbf{x})|^2$$

- Model: Brightness constancy, image shift

$$f(\mathbf{x} - \mathbf{d}) = g(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{N}$$

Taxonomy: KLT

- Algorithm
  - local linearization (Taylor expansion) $\nabla f = \left[\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\right]^T$

$$f(\mathbf{x} - \mathbf{d}) \approx f(\mathbf{x}) - \mathbf{d}^T \nabla f(\mathbf{x})$$

  - iterative solution of normal equations (Gauss-Newton)

$$\mathbf{Td} = \mathbf{r}$$

  - **T**: structure tensor (orientation tensor from outer product of gradients)

Taxonomy: KLT

- Algorithm
  - local linearization (Taylor expansion) $\nabla f = \left[\frac{\partial f}{\partial x} \; \frac{\partial f}{\partial y}\right]^T$
    $$f(\mathbf{x} - \mathbf{d}) \approx f(\mathbf{x}) - \mathbf{d}^T \nabla f(\mathbf{x})$$
  - iterative solution of normal equations (Gauss-Newton)
    $$\mathbf{Td} = \mathbf{r}$$
  - **T**: structure tensor (orientation tensor from outer product of gradients)
- C.f. block matching : different algorithm, but cost function and model can be the same.

Regularization and MAP

- In Maximum a-posteriori (MAP), the objective (or loss) ε consists of a data term and a prior

$$\min_{\mathbf{d}} \varepsilon_{\mathrm{data}}(f(\mathbf{d}), g) + \varepsilon_{\mathrm{prior}}(\mathbf{d})$$

$$\Leftrightarrow \max_{\mathbf{d}} \exp(-\varepsilon_{\mathrm{data}}(f(\mathbf{d}), g))\exp(-\varepsilon_{\mathrm{prior}}(\mathbf{d}))$$

$$\Leftrightarrow \max_{\mathbf{d}} P(g|\mathbf{d})P(\mathbf{d})$$

$$\Leftrightarrow \max_{\mathbf{d}} P(\mathbf{d}|g)$$

- A common prior is a smoothness constraint

MAP Example: KLT

- Assume a prior probability for the displacement : $P(\mathbf{d})$ (e.g. from a motion model)
- In logarithmic domain, we now have two terms in the cost function:

$$\varepsilon(\mathbf{d}) = \sum_{\mathbf{x} \in \mathcal{N}} w(\mathbf{x}) |f(\mathbf{x} - \mathbf{d}) - g(\mathbf{x})|^2 + \lambda \|\mathbf{d} - \mathbf{d}_{\text{pred}}\|^2$$

  – The standard KLT term

  – A term that *drags* the solution towards the predicted displacement (cf. Kalman filtering)

Demo: KLT

Image Reconstruction

- Assume that $\mathbf{f}$ is an unknown image that is observed through the linear operator $\mathbf{G}$: $\mathbf{f}_0 = \mathbf{G}\mathbf{f}$ + noise
- Example: blurring, linear projection
- Goal is to minimize the error $\mathbf{f}_0 - \mathbf{G}\mathbf{f}$
- Example: squared error
- Assume that we have a prior probability for the image: $P(\mathbf{f})$
- Example: we assume that the image should be smooth (small gradients)

Image Reconstruction

- Minimizing

$$\varepsilon(\mathbf{f}) = \frac{1}{2}(|\mathbf{G}\mathbf{f} - \mathbf{f}_0|^2 + \lambda(|\mathbf{D}_x\mathbf{f}|^2 + |\mathbf{D}_y\mathbf{f}|^2))$$

- Gives the normal equations

$$\mathbf{G}^T\mathbf{G}\mathbf{f} - \mathbf{G}^T\mathbf{f}_0 + \lambda(\mathbf{D}_x^T\mathbf{D}_x\mathbf{f} + \mathbf{D}_y^T\mathbf{D}_y\mathbf{f}) = 0$$

- Such that

$$\mathbf{f} = (\mathbf{G}^T\mathbf{G} + \lambda(\mathbf{D}_x^T\mathbf{D}_x + \mathbf{D}_y^T\mathbf{D}_y))^{-1}\mathbf{G}^T\mathbf{f}_0$$

- Note that often u is used for the unknown image

Gradient Operators

- Taylor expansion of image gives

$$u(x+h,y) = u(x,y) + hu_x(x,y) + O(h^2)$$

$$u(x-h,y) = u(x,y) - hu_x(x,y) + O(h^2)$$

- Finite left/right differences give

$$\partial_x^+ u = \frac{u(x+h,y) - u(x,y)}{h} + O(h^2)$$

$$\partial_x^- u = \frac{u(x,y) - u(x-h,y)}{h} + O(h^2)$$

- Often needed: products of derivative operators

Gradient Operators

- Squaring left (right) difference $(\partial_x^+)^2 u$ gives linear error in *h*
- Squaring central difference $\frac{u(x+h,y)-u(x-h,y)}{2h}$ gives quadratic error in *h*, but leaves out every second sample
- Multiplying left and right difference

$$\partial_x^+ \partial_x^- u = \frac{u(x+h,y) - 2u(x,y) + (x-h,y)}{h^2} = \Delta_x u$$

gives quadratic error in *h* (usual discrete Laplace operator)

Demo: Image Reconstruction

- IRdemo.m

Robust error norms

- A complement to RANSAC
- Assume quadratic error: *influence* of change f to $f+\partial f$ to the estimate is linear (why?)
- Result on set of measurements: mean
- Assume absolute error: influence of change is constant (why?)
- Result on set of measurements: median
- In general: sub-linear influence leads to robust estimates, but *non-linear*

Smoothness

- Quadratic smoothness term: influence linear with height of edge
- Total variation smoothness (absolute value of gradient): influence constant
- With quadratic measurement error: Rudin-Osher-Fatemi (ROF) model (Physica D, 1992)

$$\min_{u \in X} \frac{\|u - g\|^2}{2\lambda} + \sum_{1 \leq i, j \leq N} |(\nabla u)_{i,j}|$$

Total Variation (TV)

- Minimizing $$\min_{u \in X} \frac{\|u-g\|^2}{2\lambda} + \sum_{1 \le i,j \le N} |(\nabla u)_{i,j}|$$
- Stationary point
$$u - g - \lambda \operatorname{div}\left(\frac{\nabla u}{|\nabla u|}\right) = 0$$
- Steepest descent

$$u^{(s+1)} = u^{(s)} - \alpha \left(u^{(s)} - g - \lambda \frac{u_{xx}u_y^2 - 2u_{xy}u_x u_y + u_{yy}u_x^2}{|\nabla u|^3} \right)$$

Efficient TV Algorithms

- In 1D: Chambolle's algorithm (JMIV, 2004)
- In 2D:
  - Alternating direction method of multipliers (ADMM, variant of augmented Lagrangian): Split Bregman by Goldstein & Osher (SIAM 2009)
  - Based on threshold Landweber: Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) by Beck & Teboulle (SIAM 2009)
  - Based on Lagrange multipliers: Primal Dual Algorithm by Chambolle & Pock (JMIV 2011)

Demo: TV Image Denoising

TV Image Inpainting / Convex Optimization

- Note that many problems (including quadratic and TV) are convex optimization problems
- A good first approach is to map these problems to a standard solver, e.g. CVXPY by S. Diamond and S. Boyd
- Example: minimize the total variation of an image

$$\sum_{1 \leq i,j \leq N} |(\nabla u)_{i,j}|$$

under the constraint of a subset of known image values u

```
prob=Problem(Minimize(tv(X)),[X[known] == MG[known]])
opt_val = prob.solve()
```

Demo: TV Inpainting

Algorithmic Taxonomy

- Minimal problems (e.g. 5 point algorithm)
  - Fully determined solution(s)
  - Analytic solvers (polynomials, Gröbner bases)
  - Numerical methods (Dogleg, Newton-Raphson)
- Overdetermined problems (e.g. OF,BA)
  - Minimization problem
  - Numerical solvers only
  - Levenberg-Marquardt (interpolation Gauss-Newton and gradient descent / trust region)

Non-linear LS, Dog Leg

- For comparison: LM $\mathbf{r}(\boldsymbol{\beta}+\boldsymbol{\delta}) \approx \mathbf{r}(\boldsymbol{\beta}) + \mathbf{J}\boldsymbol{\delta}$

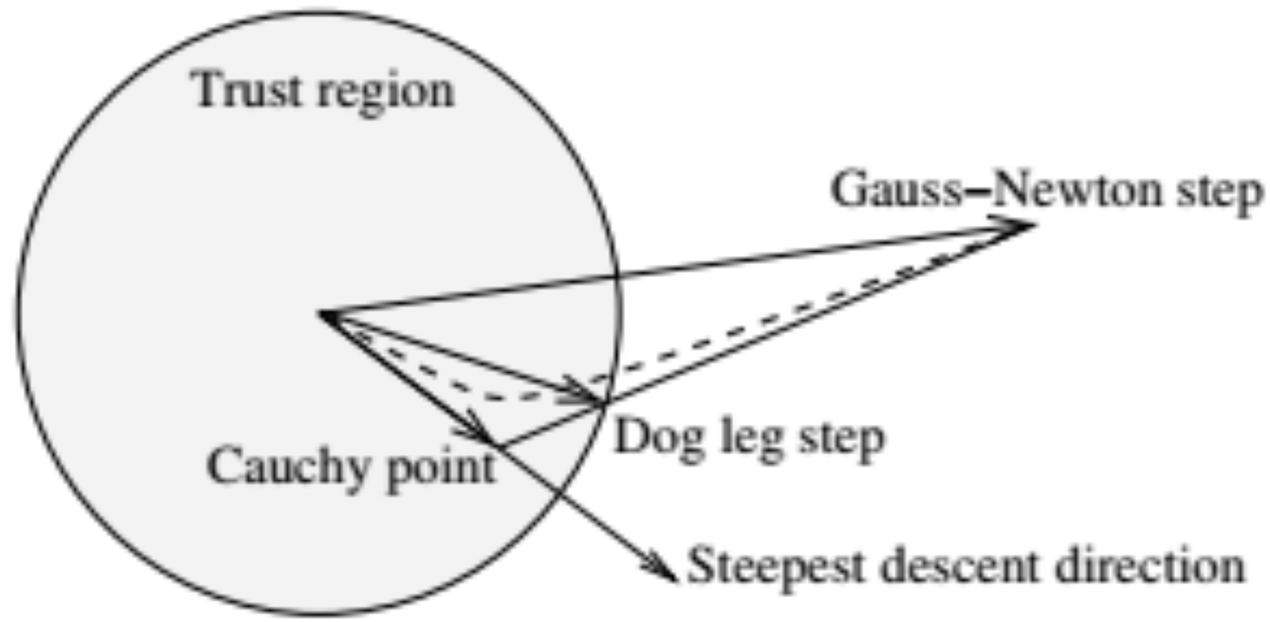
$$(\mathbf{J}^T\mathbf{J} + \lambda\,\mathbf{diag}(\mathbf{J}^T\mathbf{J}))\,\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}(\boldsymbol{\beta})$$

$$\beta_j \mapsto \beta_j + \delta_j \qquad J_{ij} = \frac{\partial r_i}{\partial \beta_j}$$

- More efficient: replace damping factor λ with trust region radius Δ

method	abbr.	properties
steepest descent	SD	$\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}$
Gauss-Newton	GN	$\mathbf{J}^T\mathbf{J}\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}$
Levenberg-Marquart	LM	combines SD and GN by damping factor
Dog Leg	DL	combines SD and GN by trust region radius Δ

Dog Leg



1. initialize Δ =1
2. compute gain factor

3. if gain factor >0

$$\boldsymbol{\beta}_{\text{new}} = \boldsymbol{\beta} + \boldsymbol{\delta}_{\text{SD}} + \alpha(\boldsymbol{\delta}_{\text{GN}} - \boldsymbol{\delta}_{\text{SD}})$$

$$\|\boldsymbol{\delta}_{\text{SD}}\| \leq \Delta \,, \quad 0 \leq \alpha \leq 1 \,, \quad \|\boldsymbol{\delta}_{DL}\| = \Delta$$

4. update gain factor
5. if update and residual nonzero goto 3

Optical Flow

- Minimizing (lecture 4) $\varepsilon(\mathbf{v}_h) = \sum_{\mathcal{R}} w |[\nabla^T f \ f_t] \mathbf{v}_h|^2$

- Under the constraint $|\mathbf{v}_h|^2 = 1$
- Using Lagrangian multiplier leads to the minimization problem

$$\varepsilon_T(\mathbf{v}_h, \lambda) = \varepsilon(\mathbf{v}_h) + \lambda(1 - |\mathbf{v}_h|^2)$$

- This is the *total least squares* formulation to determine the flow

Optical Flow

- Solution is given by the eigenvalue problem

$$\left(\sum_{\mathcal{R}} w \begin{bmatrix} \nabla f \\ f_t \end{bmatrix} \left[\nabla^T f \;\; f_t \right] \right) \mathbf{v}_h = \lambda \mathbf{v}_h$$

$$\mathbf{T}\mathbf{v}_h = \lambda \mathbf{v}_h$$

- The matrix term **T** is the spatio-temporal structure tensor
- The eigenvector with the smallest eigenvalue is the solution (up to normalization of homogeneous element)

Optical Flow

- *Local* flow estimation
  - Design question: w and R
  - Aperture problem: motion at linear structures can only be estimated in normal direction (underdetermined)
  - Infilling limited
- *Global* flow instead

Optical Flow

- Minimizing BCCE over the whole image with additional smoothness term

$$\varepsilon(\mathbf{f}) = \frac{1}{2}\int_{\Omega} (\langle \mathbf{f} | \nabla g \rangle + g_t)^2 + \lambda(|\nabla f_1|^2 + |\nabla f_2|^2)\, dx\, dy$$

- Gives the iterative Horn & Schunck method (details will follow in the lecture on variational methods)

$$\mathbf{f}^{(s+1)} = \bar{\mathbf{f}}^{(s)} - \frac{1}{\lambda^2 + |\nabla g|^2} (\langle \bar{\mathbf{f}}^{(s)} | \nabla g \rangle + g_t) \nabla g$$

Graph Algorithms

- All examples so far: vectors as solutions, i.e. finite set of (pseudo) continuous values
- Now: discrete (and binary) values
- Directly related to (labeled) graph-based optimization
- In probabilistic modeling (on regular grid): Markov random fields

Graphs

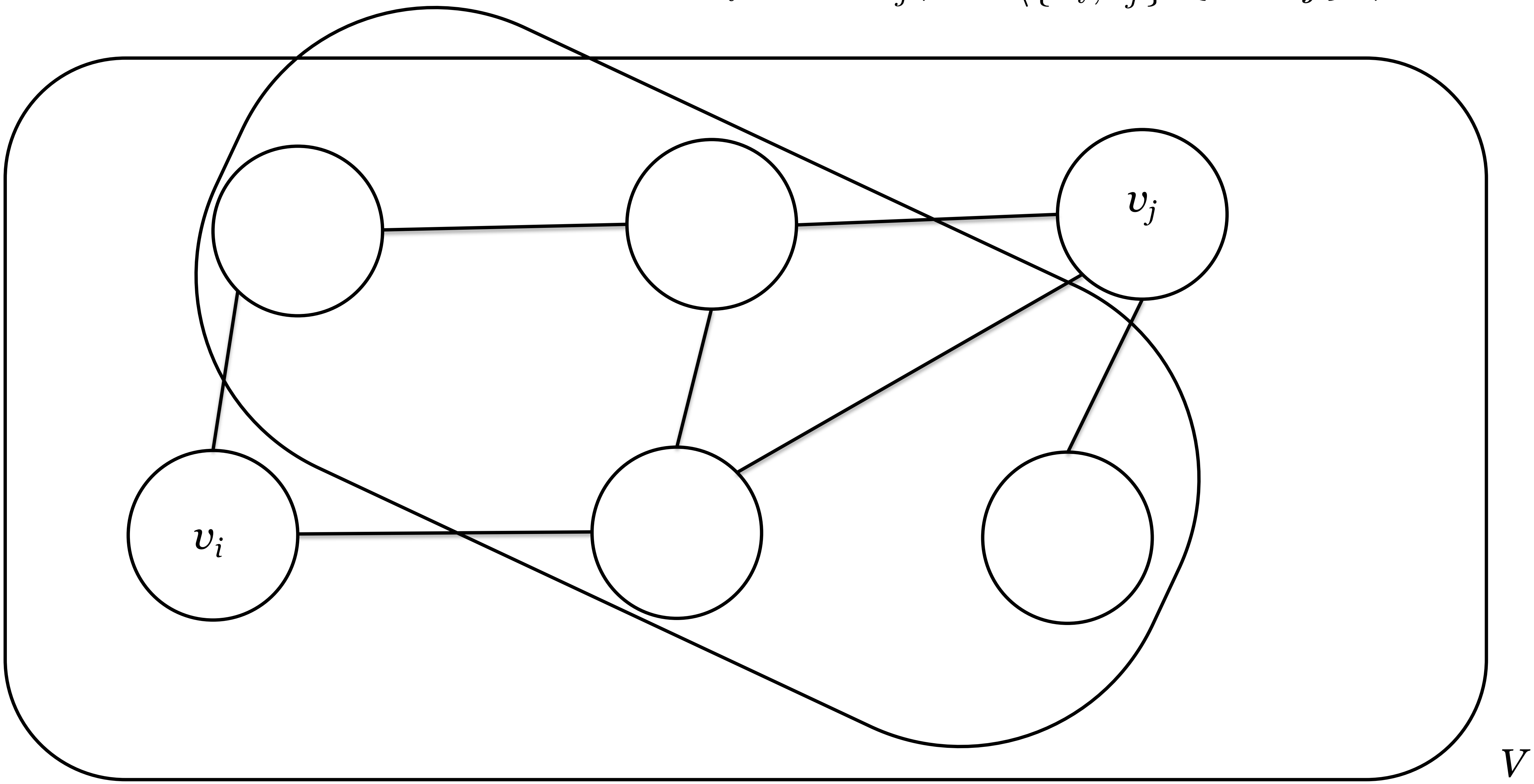
- Graph: algebraic structure $G=(V, E)$
- Nodes $V=\{v_1,v_2,\ldots,v_n\}$
- Arcs $E=\{e_1,e_2,\ldots,e_m\}$, where e_k is incident to
  - an unordered pair of nodes $\{v_i,v_j\}$
  - an ordered pair of nodes (v_i,v_j) (directed graph)
  - degree of node: number of incident arcs
- Weighted graph: costs assigned to nodes or arcs

Terminology

- Markov chain: memoryless process with r.v. X
- Markov random field (undirected graphical model): random variables (e.g. labels) over nodes with Markov property (conditional independence)
  - Pairwise $X_{v_i} \perp\!\!\!\perp X_{v_j} | X_{V \setminus \{v_i, v_j\}}$ $\{v_i, v_j\} \notin E$
  - Local $X_v \perp\!\!\!\perp X_{V \setminus (\{v\} \cup N(v))} | X_{N(v)}$
  - Global $X_A \perp\!\!\!\perp X_B | X_S$ where every path from a node in A to node in B passes through S

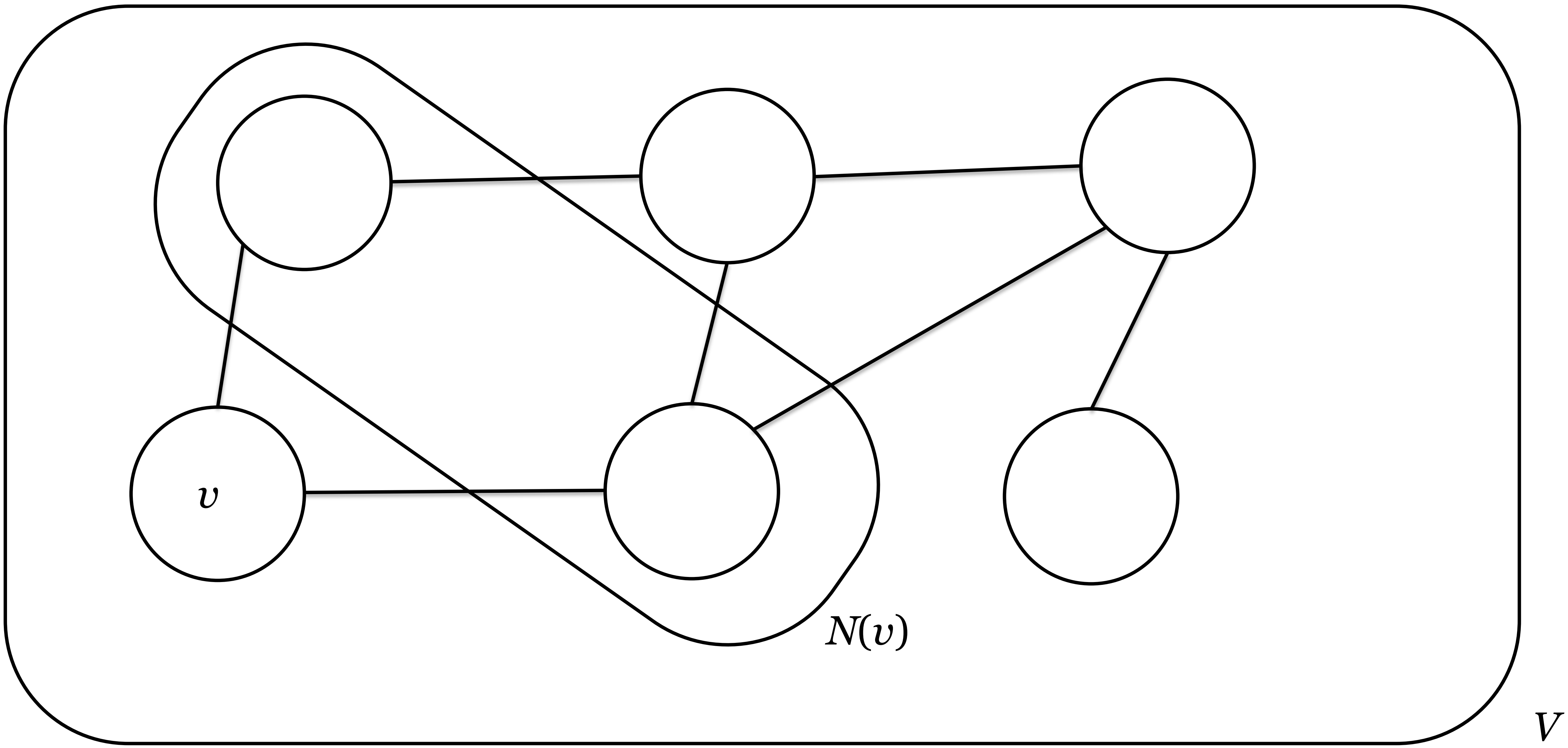
Conditional Independence

$$X_{v_i} \perp\!\!\!\perp X_{v_j} | X_{V \setminus \{v_i, v_j\}} \quad \{v_i, v_j\} \notin E$$



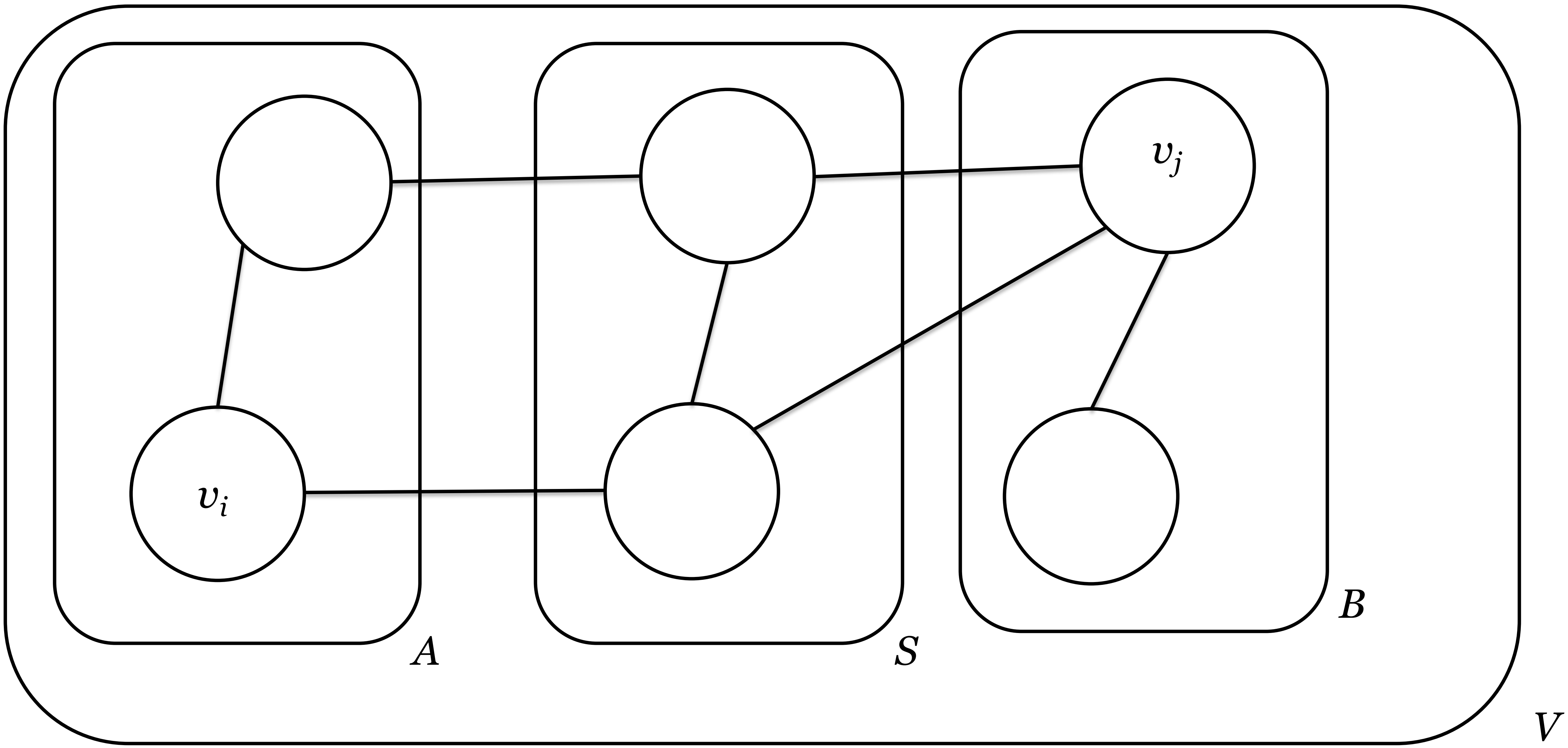
Conditional Independence

$$X_v \perp\!\!\!\perp X_{V\setminus(\{v\}\cup N(v))} | X_{N(v)}$$



Conditional Independence

$$X_A \perp\!\!\!\perp X_B | X_S$$



Terminology

- If joint density strictly positive: Gibbs RF
- Ising model (interacting magnetic spins), energy given as Hamiltonian function

$$\varepsilon(X_V) = - \sum_{e_k=\{v_i,v_j\}\in E} J_{e_k} X_{v_i} X_{v_j} - \sum_{v_j} h_{v_j} X_{v_j}$$

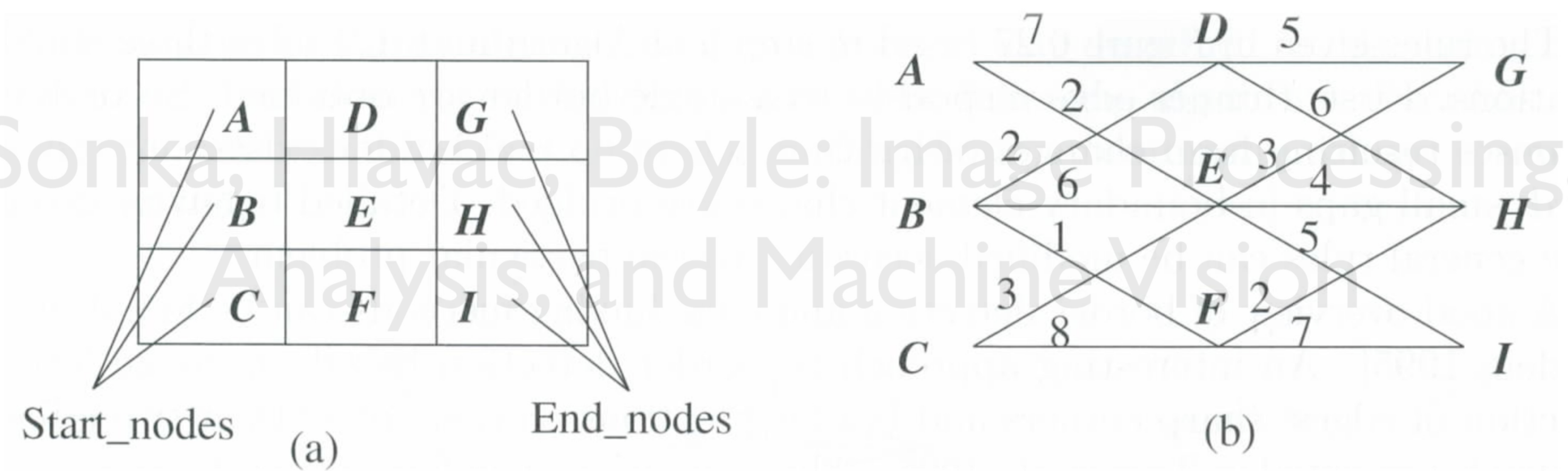
- General form

$$\varepsilon(X_V) = \lambda \sum_{e_k=\{v_i,v_j\}\in E} V(X_{v_i}, X_{v_j}) + \sum_{v_j} D(X_{v_j})$$

- Configuration probability $P(X_V) \propto \exp(-\varepsilon(X_V))$

1D: Dynamic Programming

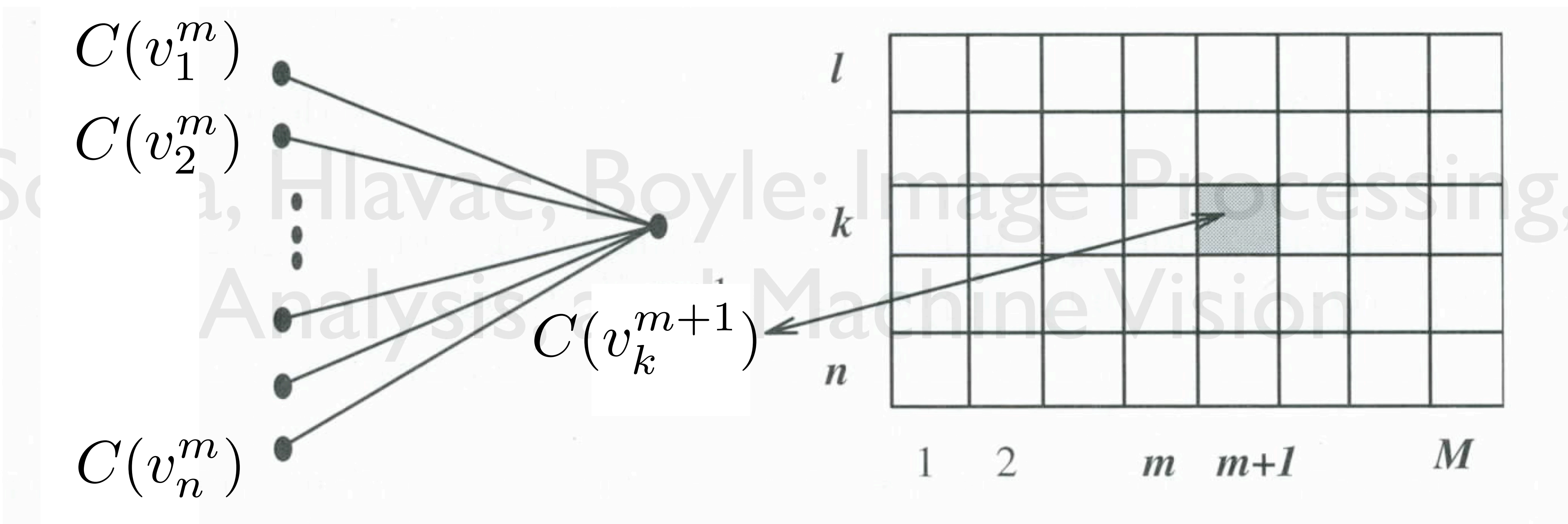
- Problem: find optimal path from source node *s* to sink note *t*
- Principle of Optimality: If the optimal path *s*-*t* goes through *r*, then both *s*-*r* and *r*-*t*, are also optimal



(a) (b)

1D: Dynamic Programming

- $C(v_k^{m+1})$ is the new cost assigned to node v_k
- $g^m(i,k)$ is the partial path cost between nodes v_i and v_k



1D: Dynamic Programming

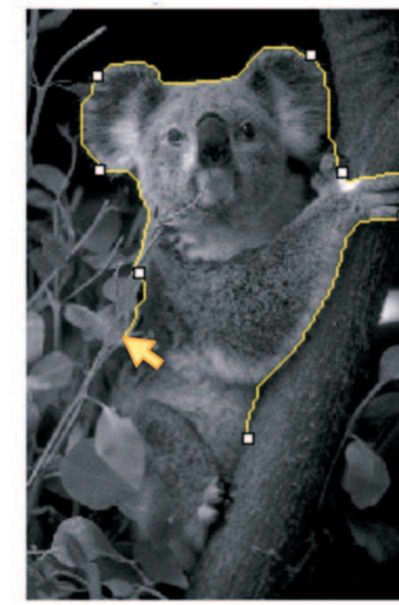
- $C(v_k^{m+1})$ is the new cost assigned to node v_k
- $g^m(i,k)$ is the partial path cost between nodes v_i and v_k

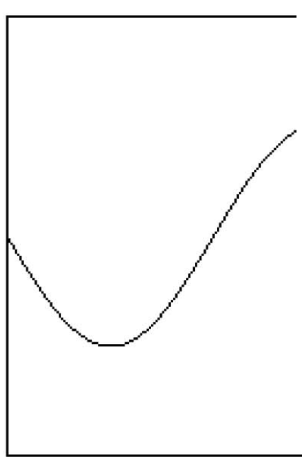
$$C(v_k^{m+1}) = \min_i \left(C(v_i^m) + g^m(i,k)\right)$$

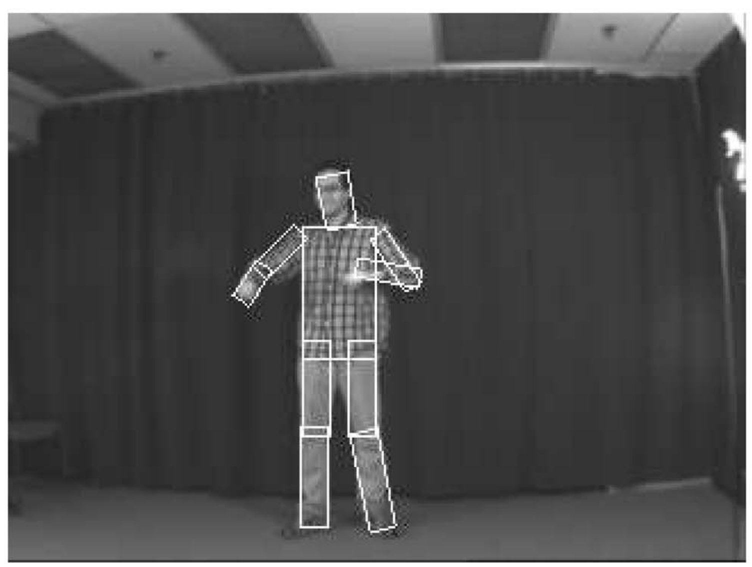
$$\min\left(C(v^1, v^2, \ldots, v^M)\right) = \min_{k=1,\ldots,n} \left(C(v_k^M)\right)$$

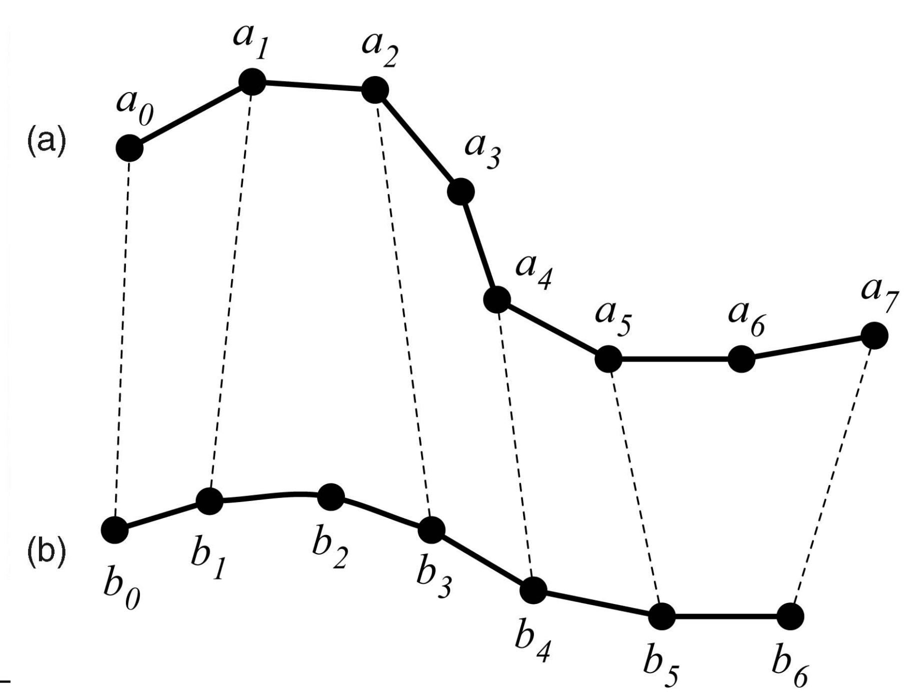
Examples

- Shortest path computation (contours / intelligent scissors)
- 1D signal restoration (denoising)
- Tree labeling (pictorial structures)
- Matching of sequences (curves)









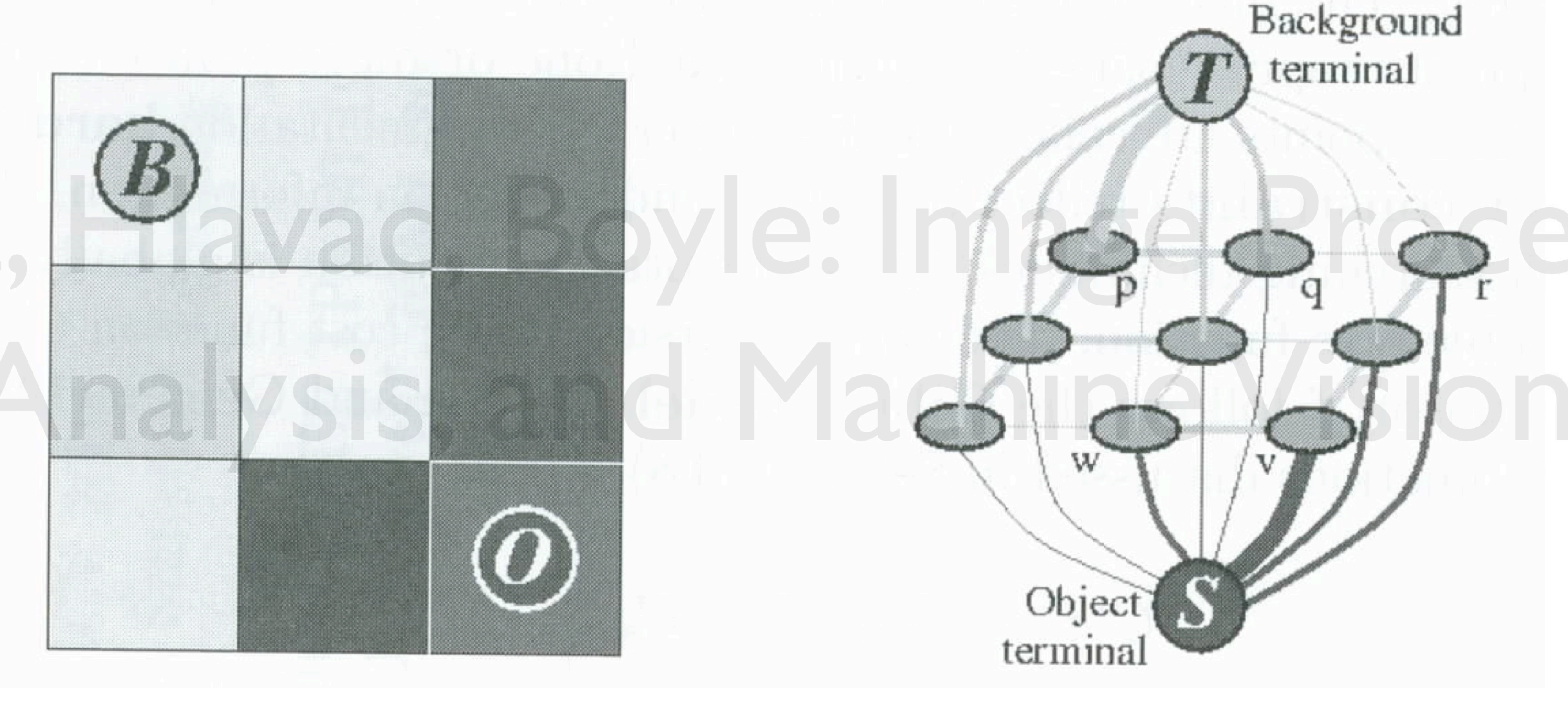
Gibbs Model /Markov Random Field

- Attempts to generalize dynamic programming to higher dimensions unsuccessful
- Minimize $C(f) = C_{\mathrm{data}}(f) + C_{\mathrm{smooth}}(f)$
  using arc-weighted graphs $G_{\mathrm{st}} = (V \cup \{s, t\}, E)$
- Two special terminal nodes, source *s* (e.g. object) and sink *t* (e.g. background) hard-linked with seed points

Graph Cut: Two types of arcs

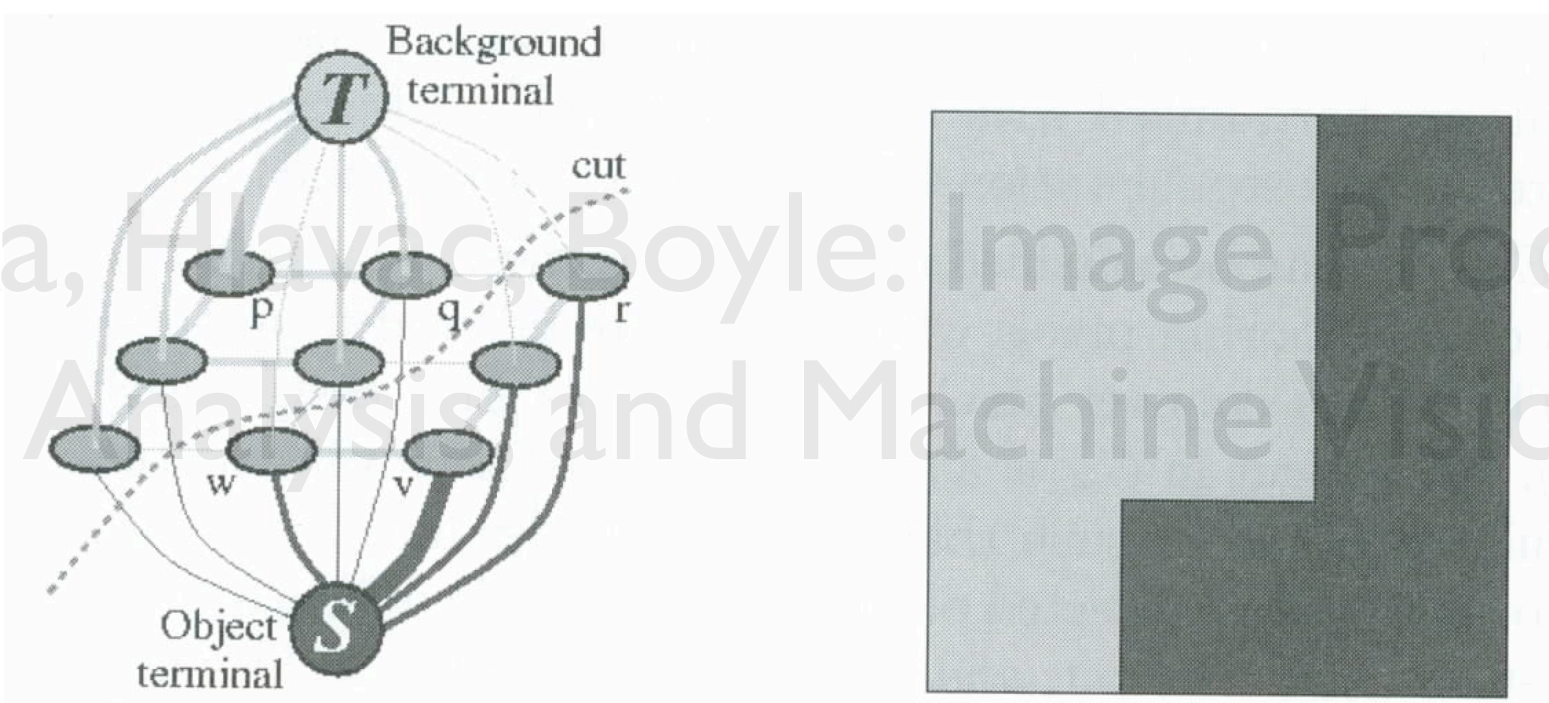
– n-links: connecting neighboring pixels, cost given by the smoothness term V

– t-links: connecting pixels and terminals, cost given by the data term D



Graph Cut

- *s*-*t* cut is a set of arcs, such that the nodes and the remaining arcs form two disjoint graphs with points sets *S* and *T*
- cost of cut: sum of arc cost
- minimum *s*-*t* cut problem (dual: maximum flow problem)



Graph Cut

- n-link costs: large if two nodes belong to same segment, e.g. inverse gradient magnitude, Gauss, Potts model
- t-link costs:
  - *K* for hard-linked seed points (*K* > maximum sum of data terms)
  - 0 for the opposite seed point
- Submodularity $V(\alpha,\alpha) + V(\beta,\beta) \leq V(\alpha,\beta) + V(\beta,\alpha)$

Demonstration

Examples / Discussion

- Binary problems solvable in polynomial time (albeit slow)
  - Binary image restoration
  - Bipartite matching (perfect assignment of graphs)
- N-ary problems (more than two terminals) are NP-hard and can only be approximated (e.g. α-expansion move)
  - Stereo application has quantization (it used to be popular because many evaluation sets used discrete depths)

Michael Felsberg
michael.felsberg@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET