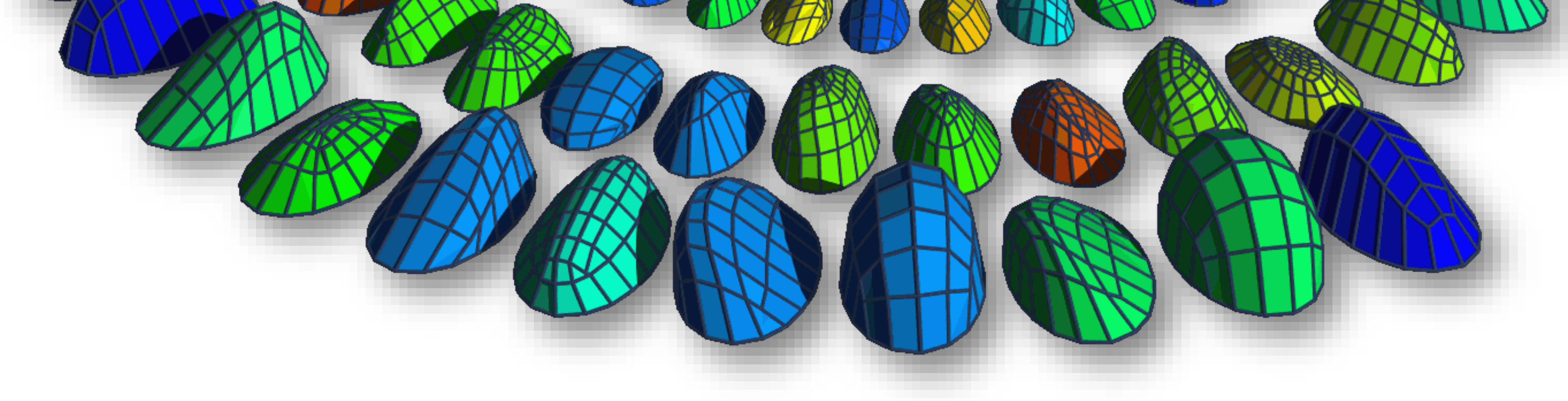


Integer Programming for Layout Problems

Peter Wonka, KAUST

SA '18 Courses, December 04-07, 2018, Tokyo, Japan
ACM 978-1-4503-6026-5/18/12.
10.1145/3277644.3277794

Linear Programming

- General Form

$$\max c^T x$$

$$Ax \leq b$$

- Example

$$\max c_1 x_1 + c_2 x_2$$

$$a_{11} x_1 + a_{12} x_2 \leq b_1$$

$$a_{21} x_1 + a_{22} x_2 \leq b_2$$

$$\max 4x_1 + 2x_2$$

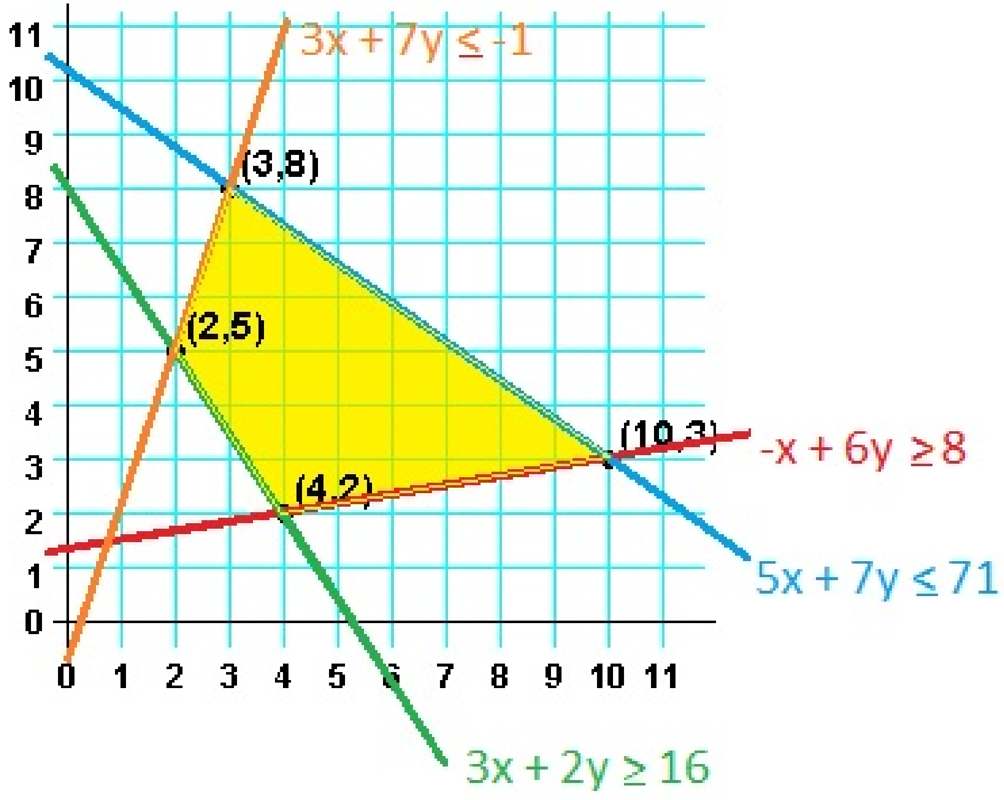
$$x_1 + 2x_2 \leq 12$$

$$7x_1 \leq 22$$

How to solve linear programming problems?

- Simplex algorithm / interior point algorithms
- Standard solvers / quite fast
- Formulation is already non-trivial

- Graphical Example



Variations

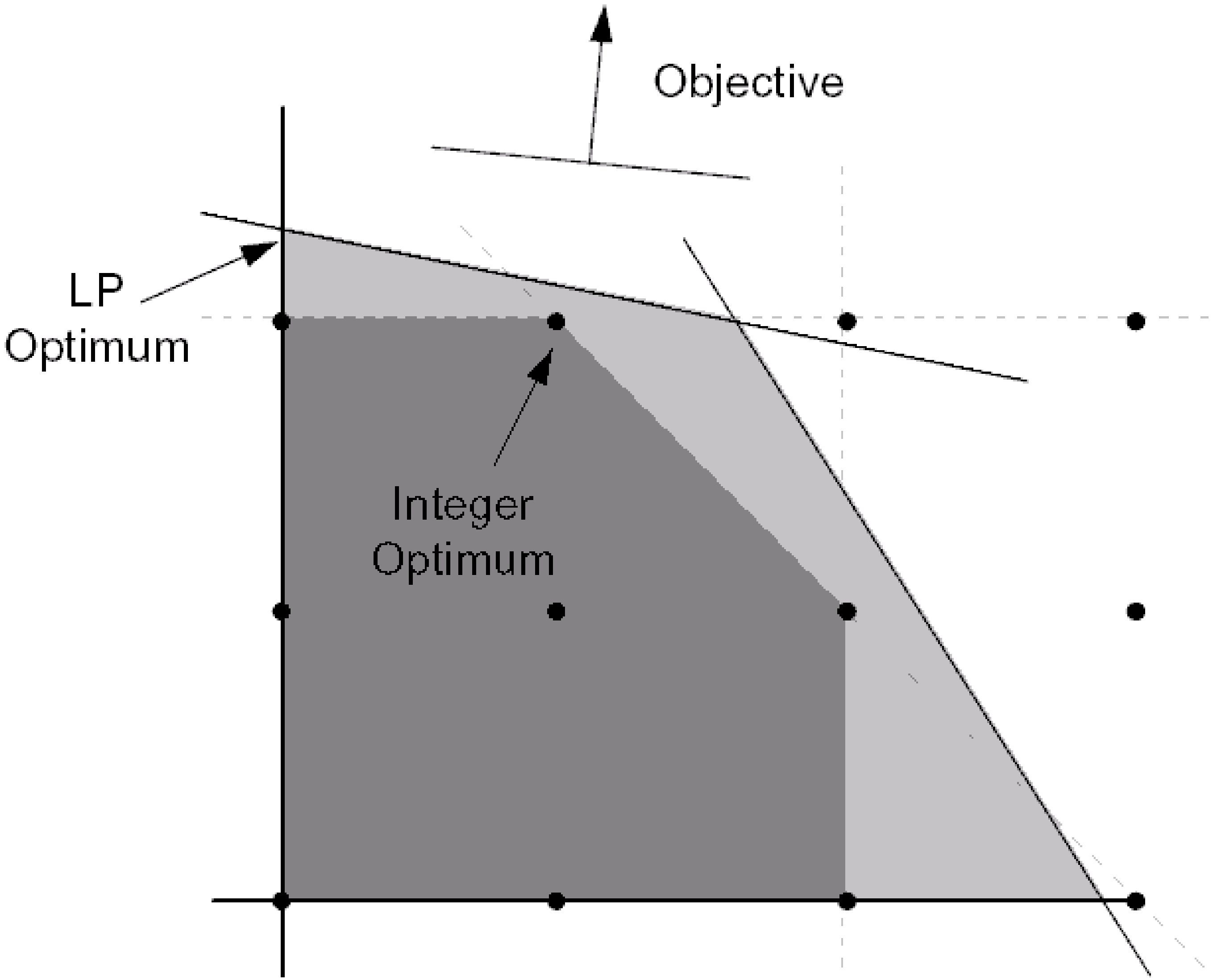
- Float variables
  → linear program (LP)
- Integer variables
  → (linear) integer program (IP)
- Float and integer variables
  → mixed integer program (MIP)
- Binary variables
  → binary integer program (BIP)

$$\max c^T x$$
$$Ax \leq b$$

- Switch **min** and **max**
- Switch $\leq, \geq, =$
- Require all variables ≥ 0
- Examples:

$$\min cx$$
$$Ax \leq b$$
$$x \geq 0$$

Graphical Example



Optimization

- Modeling:
  How to formulate an application problem as a standard optimization problem?
- Algorithm Development:
  How to derive new optimization algorithms for standard optimization problems or specialized optimization problems?
- Optimization "Theory":
  Finding convergence guarantees, bounds, … of optimization algorithms

Multiple ways to publish using optimization

good

- Modeling: propose an interesting problem formulation for a new or an existing problem
- Algorithm: propose a new algorithm for a specific formulation
- Modeling + Algorithm

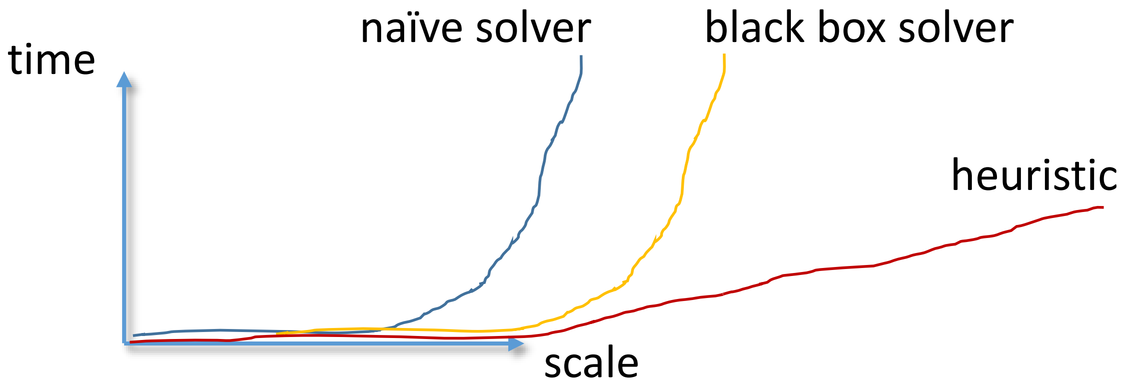
---

not so good

- Theory: doesn't work well in visual computing / can work in ML
- Do Nothing: just publish a known formulation
- Worse than nothing: publish an ad-hoc algorithm to a problem that has a known / efficient formulation
- Pretend algorithm development: pretend to develop a new algorithm while copying the derivation from another source

How to solve an IP Problem?

- (Step 1): check if the problem is difficult or a simpler special case
  - difficult means that a polynomial algorithm is unlikely to exist
- Step 2: use a black box solver such as Gurobi, matlab, …



- Step 3: choose from the options below
  - develop a new exact or heuristic algorithm for the specific problem
  - reuse an existing heuristic algorithm
  - reformulate the problem to make it easier to solve, use existing extensions / tricks

Knapsack Problem

Standard Problem

- Input:
  - a set of items i with values v_i and weights w_i
  - a knapsack with maximum capacity C

$C = 10$

$w_1 = 5$ $\quad v_1 = 3$

$w_2 = 8$ $\quad v_2 = 7$

$w_3 = 3$ $\quad v_3 = 5$

- Formulation: $x_i = 1$ means we pack item i in the knapsack

$$\max v^T x$$

$$w^T x \leq C$$

$$x_i \in \{0,1\}$$

- Difficulty: difficult in general, but DP solution exists for integer weights and capacity

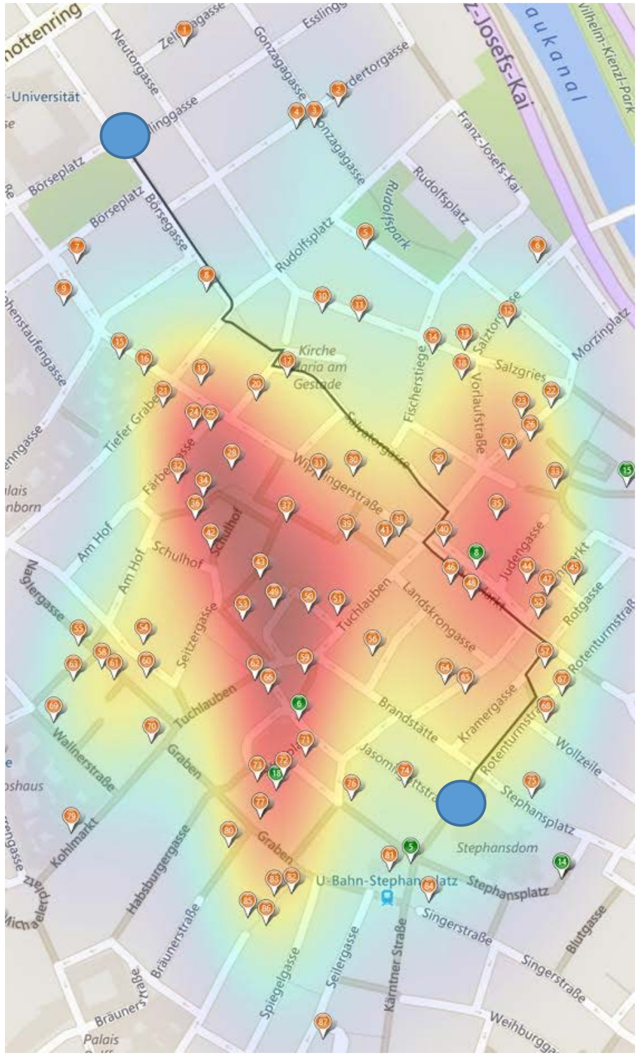
Matlab Example

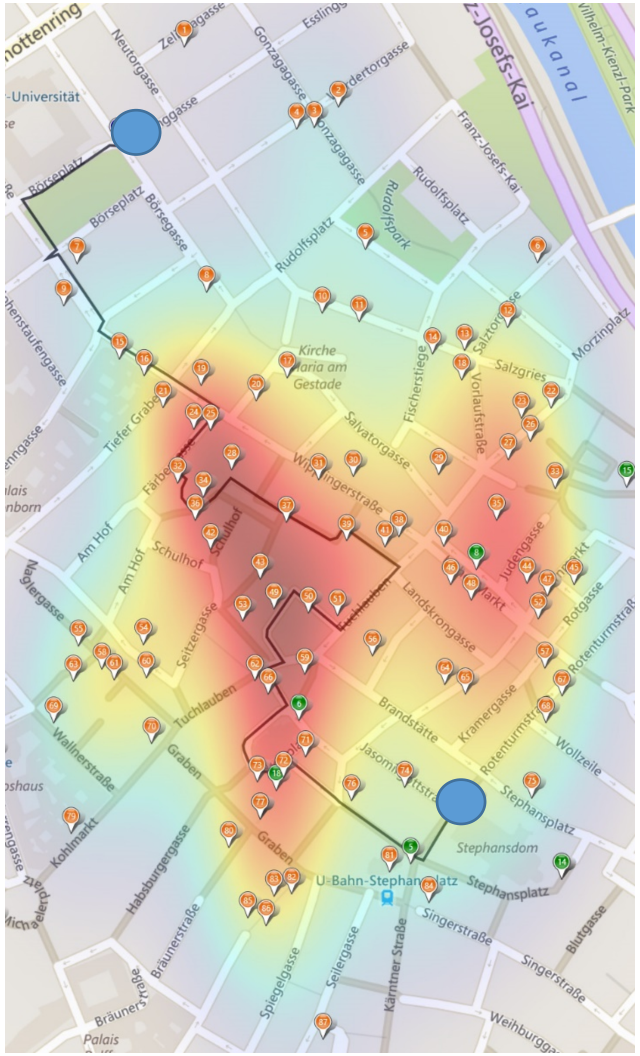
```
C = 750
weights = [ 70; 73; 77; 80; 82; 87; 90; 94; 98; 106; 110; 113; 115; 118; 120];
values = [ 135; 139; 149; 150; 156; 163; 173; 184; 192; 201; 210; 214; 221; 229;
240];

LZero = zeros(length(weights),1);
LOne = ones(length(weights),1);
LCount = 1:length(weights);


tic;
intlinprog( -values, LCount, weights', C, [], [], LZero, LOne)
toc;
```

City Exploration

TVCG 2017

- Input: city map as graph (nodes and edges)
  start and end location (node) on the map
  c – edge attractiveness for each edge
  t – time it takes to walk along an edge
  T – maximum time allowed
- Goal: find a walk through the city from start to end that explores the most worthy edges, but stays under the time limit

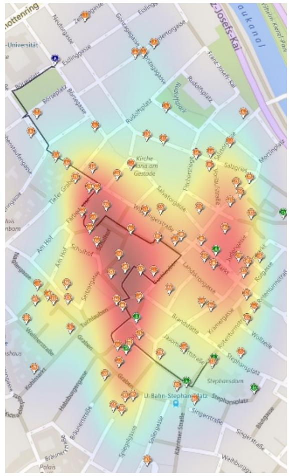




City Exploration

TVCG 2017

- Input: city map as graph (nodes and edges)
  start and end location (node) on the map
  c – edge attractiveness for each edge
  t – time it takes to walk along an edge
  T – maximum time allowed
- Goal: find a walk through the city from start to end that explores the most worthy edges, but stays under the time limit



- Variables:
  - $x_i = 1$ if edge i is selected
  - $v_j = 1$ if vertex j is selected

  $$x_i, v_j \in \{0,1\}$$
- Time constraint

  $$t^T x \leq T$$
- Connection constraint

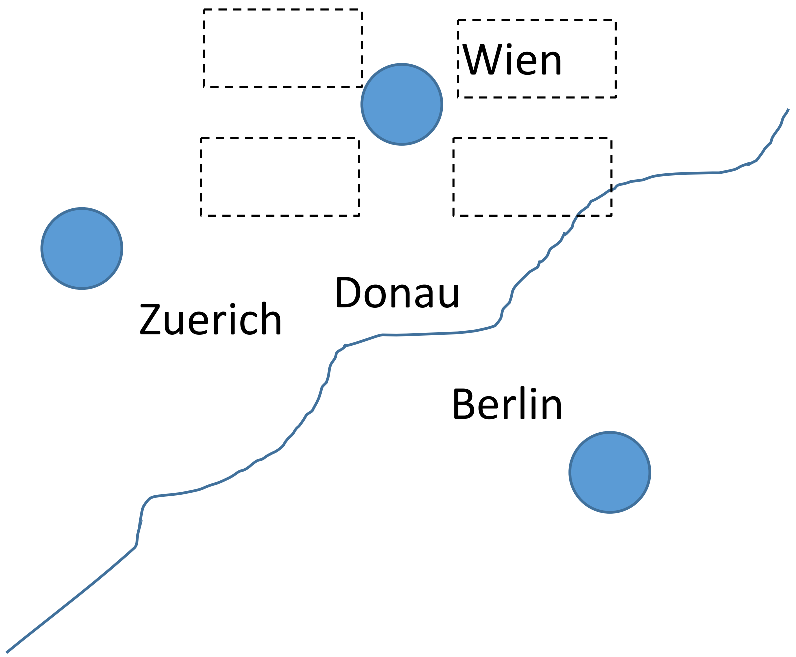
  $$\sum_{i \in N_j} x_i = 2v_j \qquad \sum_{i \in N_s} x_i = 1 \qquad \sum_{i \in N_e} x_i = 1$$
- Objective function:

  $$\max c^T x$$
- Problem: can create isolated cycles
- Solution: lazy constraint adding

Map Labeling Problem

???

- Given a set of map objects i (cities, streets, rivers, …) and corresponding labels
- Goal: place labels without overlap
- IP Formulation: discretize possible label positions j



- Variable definition: $x_{ij} = 1$ if label i is placed at position j

$$x_{ij} \in \{0,1\}$$

- Coverage: Each element is labeled exactly once:

$$\sum_j x_{ij} = 1$$

- Non-overlap for each conflicting placement

$$x_{ij} + x_{lm} \leq 1$$

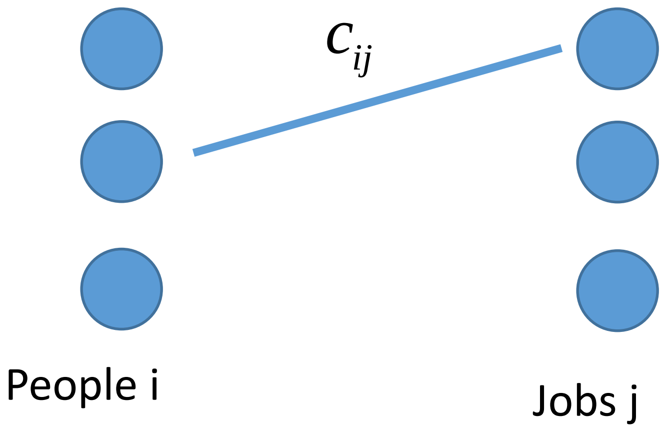
- Objective (assuming some positional preferences)

$$\min \sum_i \sum_j c_{ij} x_{ij}$$

Assignment Problem

Standard Problem

- n people carry out n jobs
- Each person i is assigned to exactly one job j
- Qualification is modeled by a cost c_{ij} for person i being assigned to job j



- Variable definition: $x_{ij} = 1$ if person i does job j
- Limited Work: Each person i does one job: for all i

$$\sum_j x_{ij} = 1$$

- Coverage: Each job is done by one person: for all j

$$\sum_i x_{ij} = 1$$

- All variables are binary, minimize cost

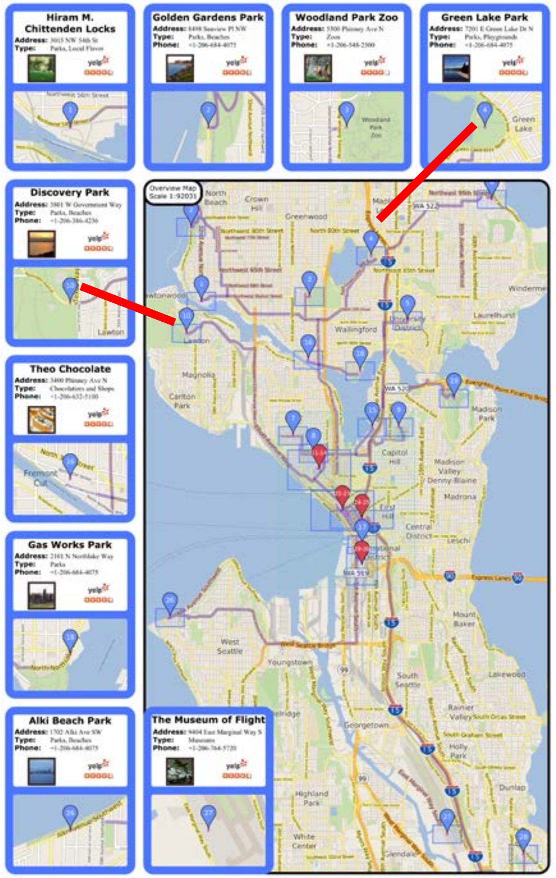
$$x_{ij} \in \{0,1\} \qquad \min \sum_i \sum_j c_{ij} x_{ij}$$

- Difficulty: specialized algorithm exists

Tourist Map Layout

EG 2014

- Overview Map
- Points of Interest (POIs)
- Detail maps for each POI
- Cost C_{ij} corresponds to the distance of POI on overview map and the detail map
- Standard assignment problem



Tourist Map Layout

EG 2014

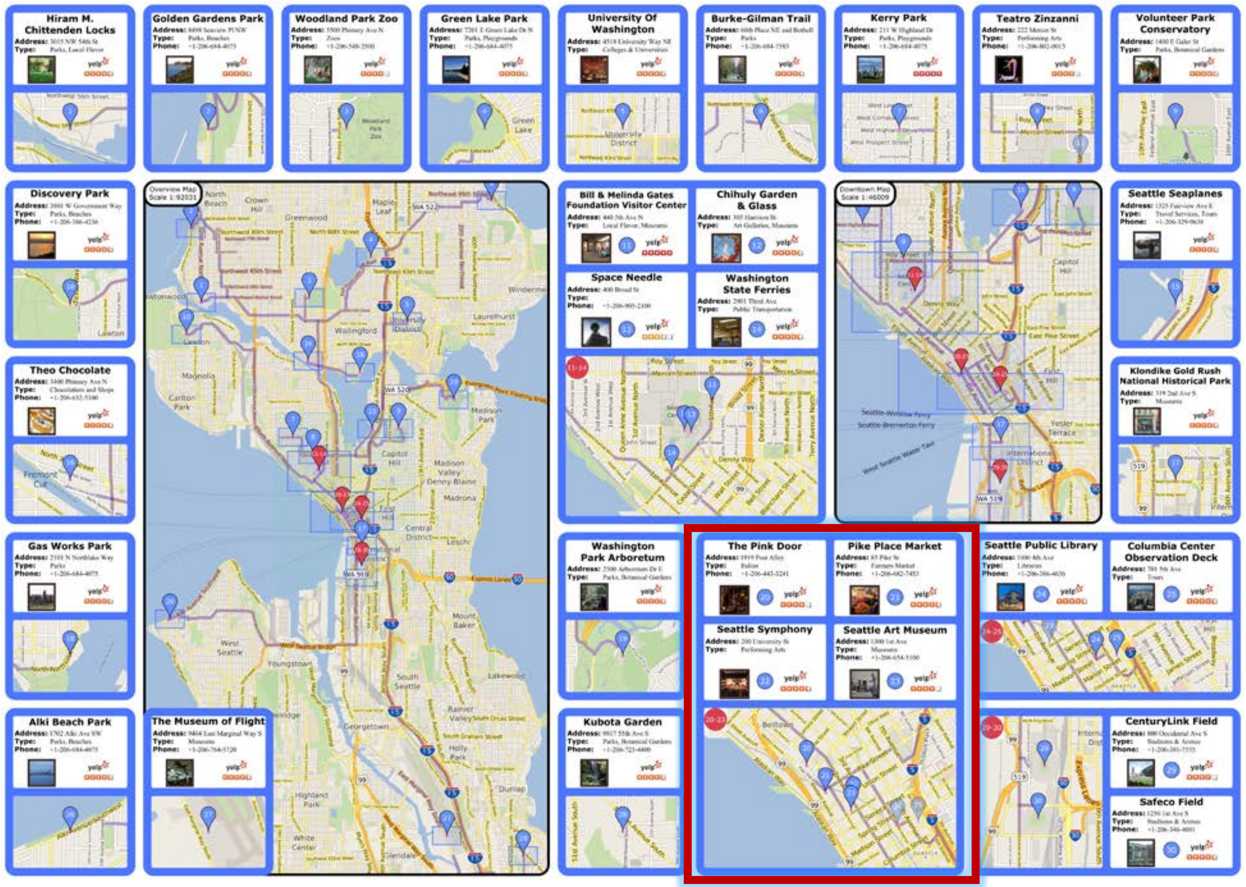
- Extension: include detail maps larger than one grid cell
- Variables $x_{ij} = 1$ if top left corner of detail map i is assigned to grid pos j
  Note: not all combinations possible
- Coverage: for each pos j

$$\sum_{(i,j) \in C_j} x_{ij} = 1$$

← placement (i,j) covers pos j

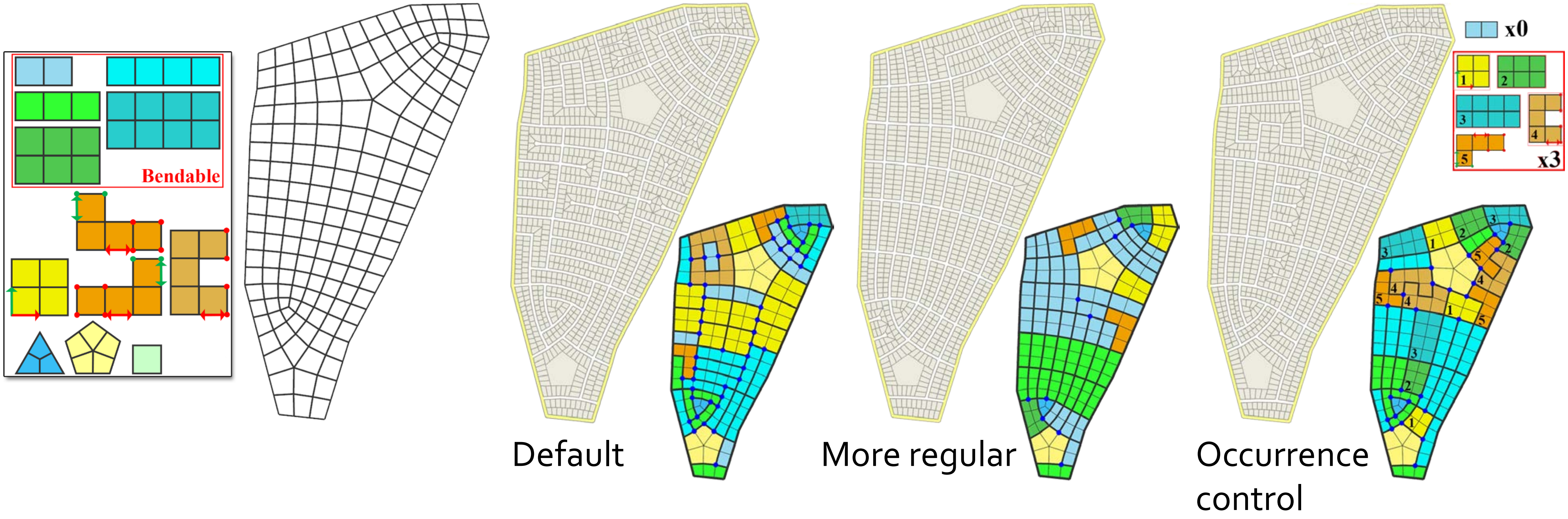
- One time placement: for each map i

$$\sum_j x_{ij} = 1$$



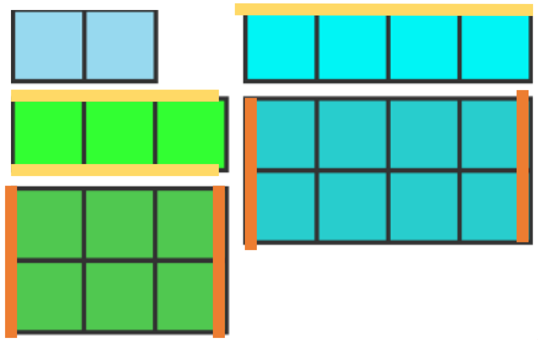
Urban Layouts

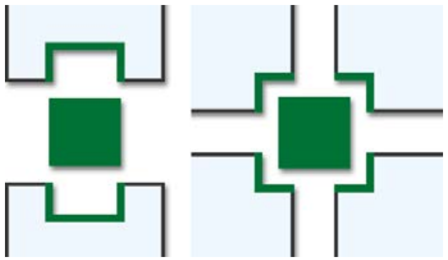
SG 2014

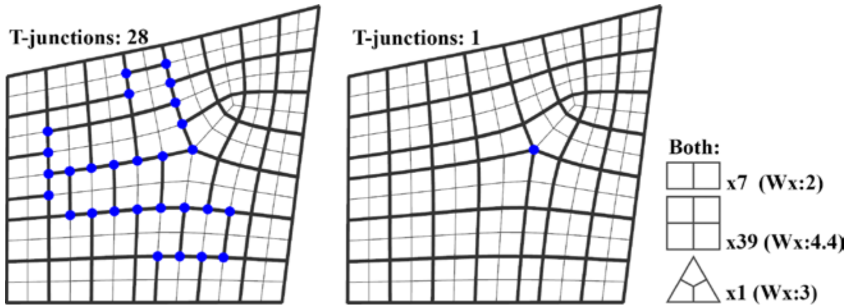


Urban Layouts

- Drop constraint that each tile can only occur once → replace it with general occurrence control
  - e.g. exactly one school tile, 2-4 store tiles, …
- Tiles can be placed in multiple orientations
- Cost is modeled by deformation cost of the regular template
- Add color constraints to enforce that only sides with matching colors can be adjacent to each other. Can be modeled as hard or soft constraint. Vertex based constraints to limit T-junctions.







Floor Planning

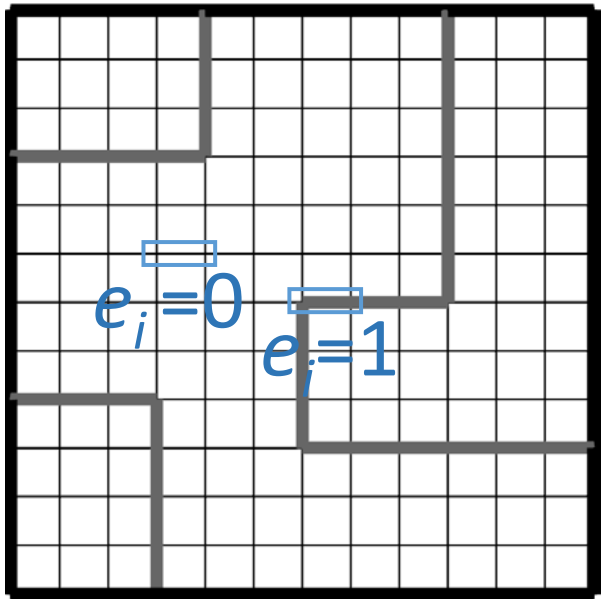
SG 2014

- Meet both **accessibility** (corridors) and **aesthetic** (room shapes) criteria of floor plans of large facilities

Network Modeling

SG 2016

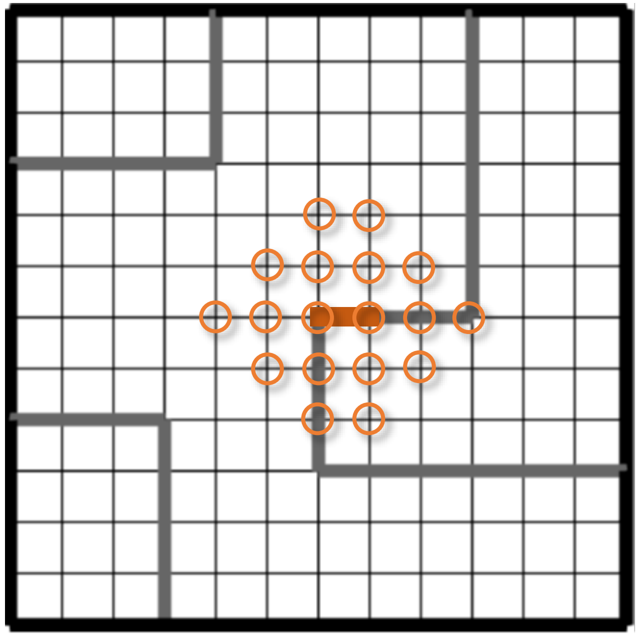
- find a subset of mesh edges that
  optimize a set of quality measures while satisfying validity constraints:
  - Coverage
  - Connectivity



Modeling Coverage Constraint

- Every vertex is within the coverage range of the network edges.

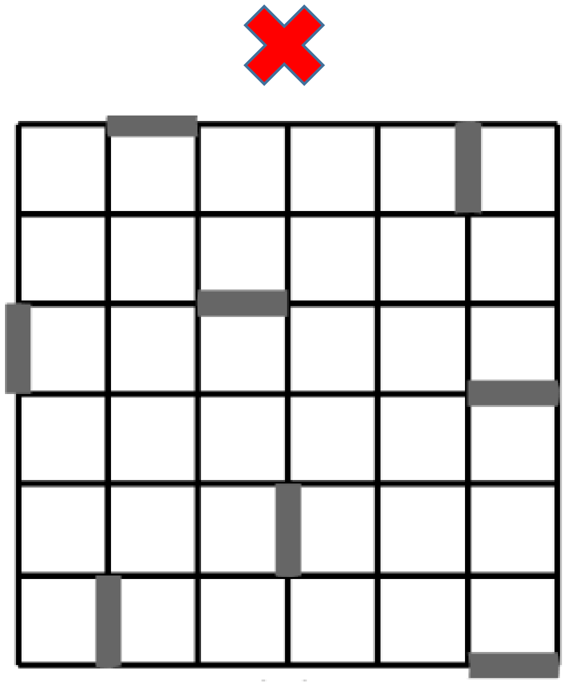
$$\forall_{v \in V} \sum_{e_i \text{ covers } v} e_i \geq 1$$



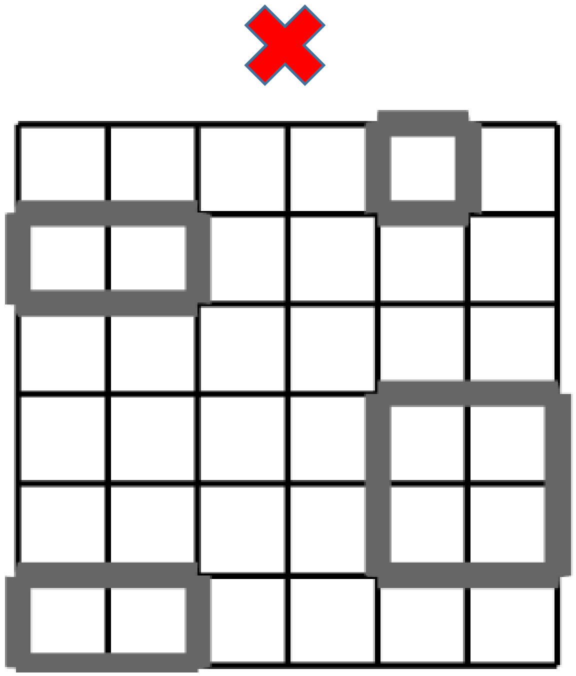
(Coverage range = 2)

Modeling Connectivity Constraint

- A global phenomenon – cannot be modelled locally.



By coverage constraint alone



Forbid dead-end vertices

$$\forall_{v \in network} \sum_{e_i \text{ touches } v} e_i \geq 2$$

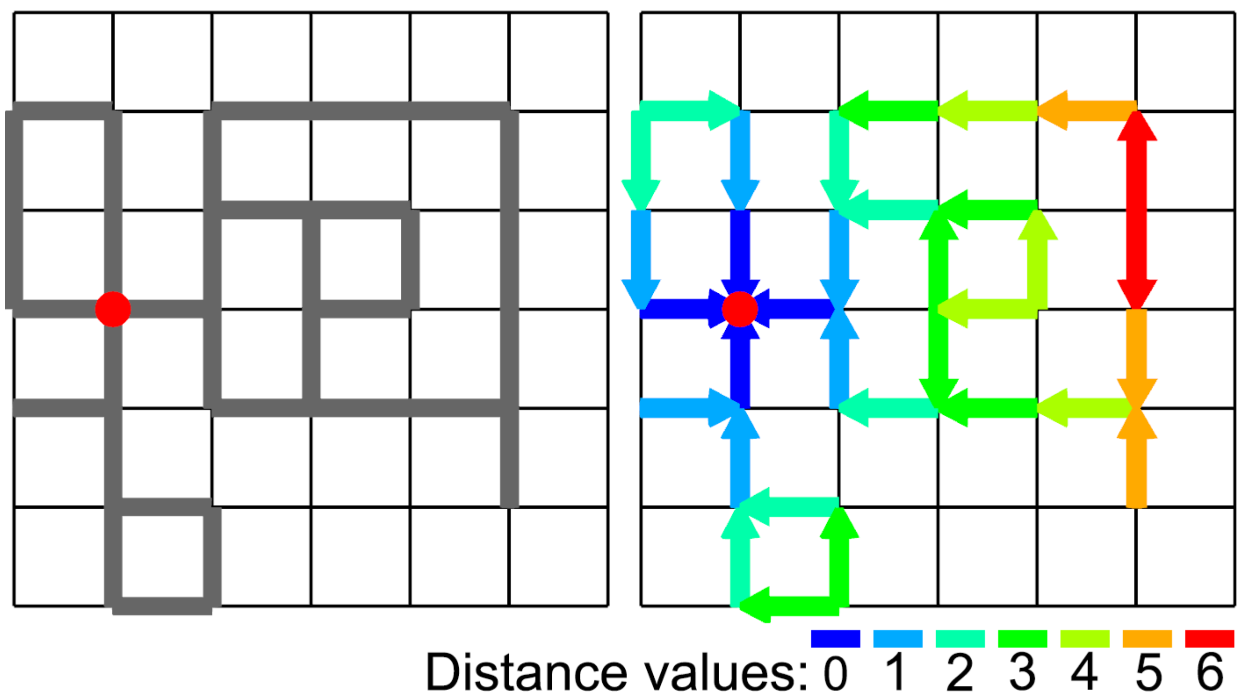
Modeling Connectivity Constraint

- Succeeding half-edges of a network must have descending "***distance***" values, except at sinks.

$$\forall_{e_{i-j} \in \text{network}} \; e_{i\to j} + e_{j\to i} > 0$$

Distance value

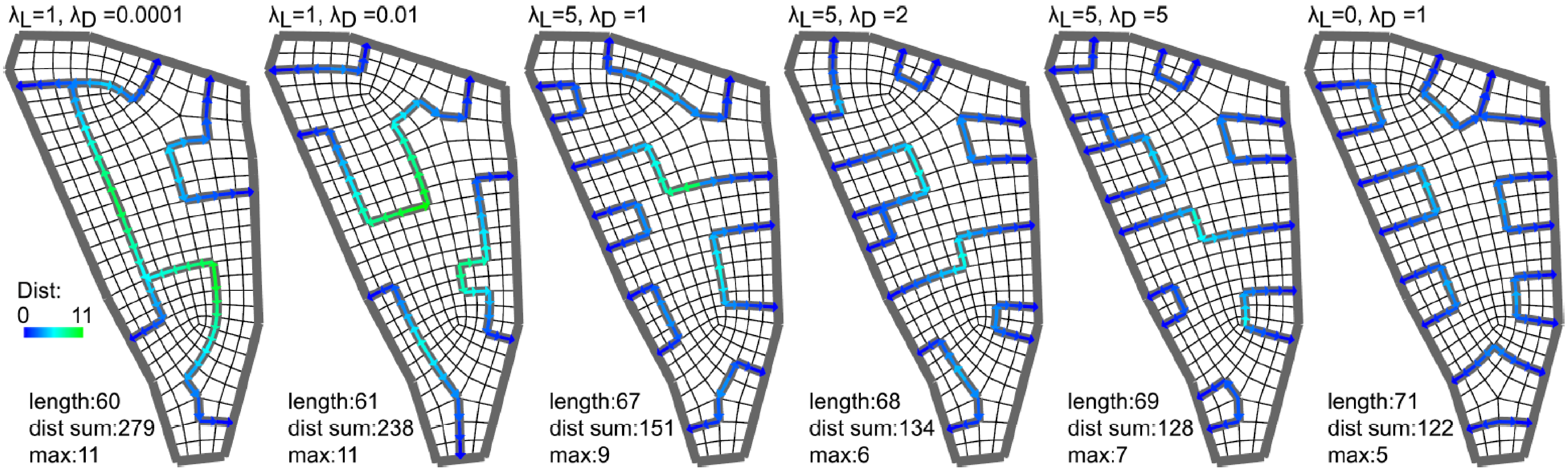
$$\forall_{\substack{e_{i\to j} \in \text{network}, \\ v_j \text{ not a sink}}} \; \exists_{e_{j\to k} \in \text{network}} \; \boxed{D_{i\to j}} > D_{j\to k}$$



Energy Function

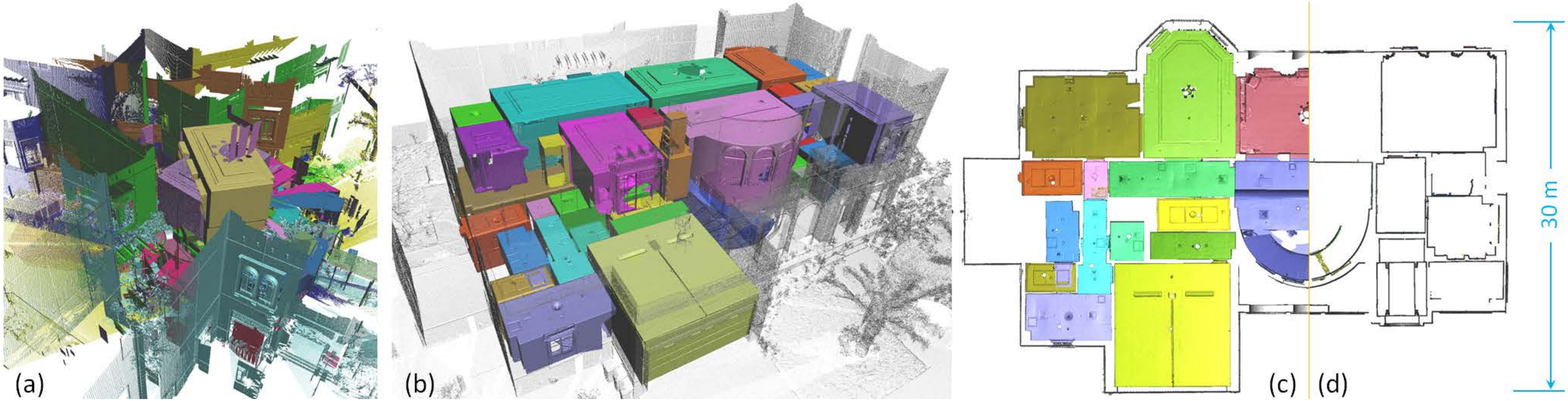
Network length | Distances to sinks

$$\text{minimize}\ \lambda_L \sum \Delta_i e_i + \lambda_D \sum D_{i\to j}$$



Scan Registration

SGA 2016

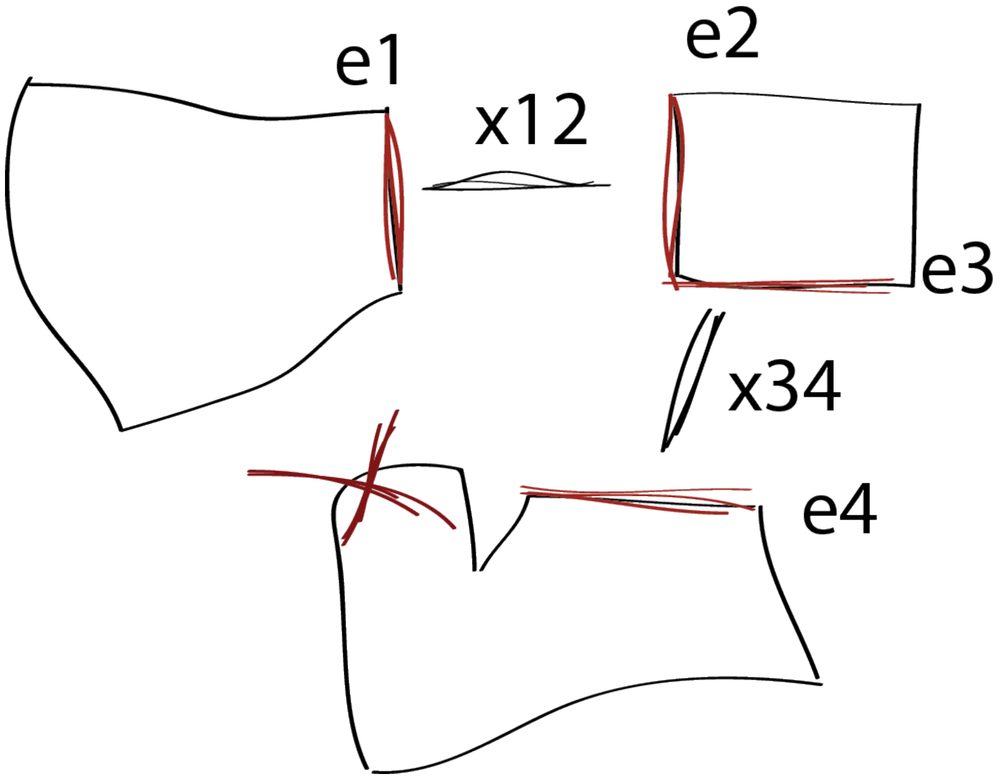


Puzzle problem

- Input: set of puzzle pieces
  f - fitness scores for matching a side of one piece to a side of another piece
- Variables: $x_{ij} = 1$ if edge i matches edge j
- Objective function: max $f^T x$
- Constraints: for every edge i:

$$\sum_j x_{ij} \leq 1$$

- Symmetry: $x_{ij} = x_{ji}$
- Intersection Avoidance:
  - add constraints on demand
  - e.g. $x_{12} + x_{34} \leq 1$



(Mixed-)Integer Quadratic Programming

- General Form

$$\max cx + x^T C x$$
$$Ax \leq b$$

Discrete MDS

EG 2015, Princeton (Quadratic Assignment)

- Input:
  - set of images; image distances d_{ij}
- Goal: assign image tiles to grid cells so that distances in the grid reflect the given distances
- Example: image distances based on optimal transport computed on color histograms
- Discretized version of MDS

Discrete MDS

EG 2015, Princeton (Quadratic Assignment)

- Input:
  - set of images; image distances d_{ij}
- Goal: assign image tiles to grid cells so that distances in the grid reflect the given distances
- Example: image distances based on optimal transport computed on color histograms
- Discretized version of MDS

- Variables: $x_{ij} = 1$ if image i is assigned to pos j
- Cost matrix C derived from d_{ij}
  Note: size #of images to the power of 4
- Coverage and Non-overlap:

$$\sum_j x_{ij} = 1 \qquad \sum_i x_{ij} = 1$$

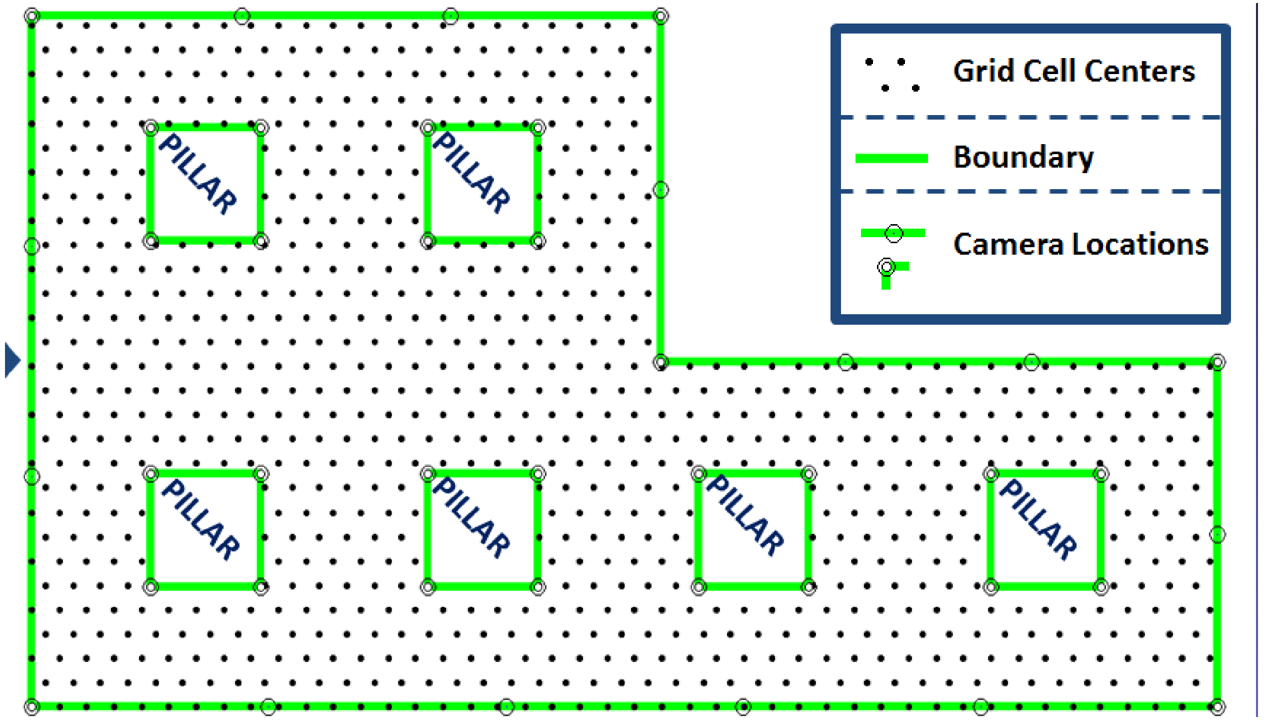
- Objective:

$$\min x^T C x$$

Camera Placement

EG 2015

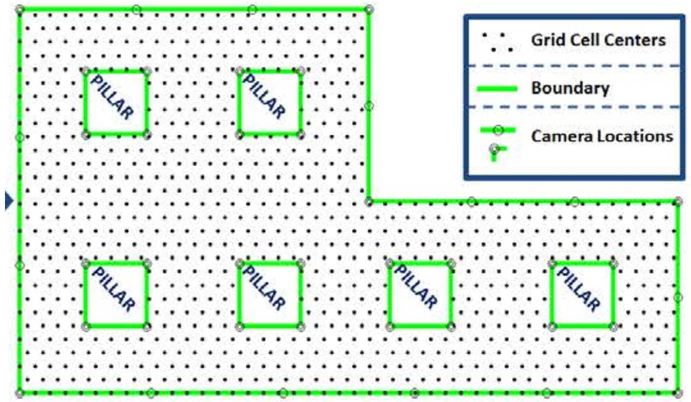
- Input
  - Room sampled into grid cells j
  - Possible camera positions l
  - $c_{lm} = 1$ cost for selecting a pair of cameras l,m
- Output
  - select a sparse set of cameras that see the room



Camera Placement

EG 2015

- Input
  - Room sampled into grid cells j
  - Possible camera positions I
  - $c_{lm} = 1$ cost for selecting a pair of cameras l,m
- Output
  - select a sparse set of cameras that see the room



L_s

- Variables: $x_i = 1$ if camera i is selected $x_i \in \{0,1\}$
- Position conflict constraints: For each location L_s

$$\sum_{i \in L_s} x_i \leq 1$$

(because multiple rotated cameras can be at L_s)

- Visibility constraint: (grid cell visibility computed by ray tracing, each column of V corresponds to one camera)

$$Vx \geq 1$$

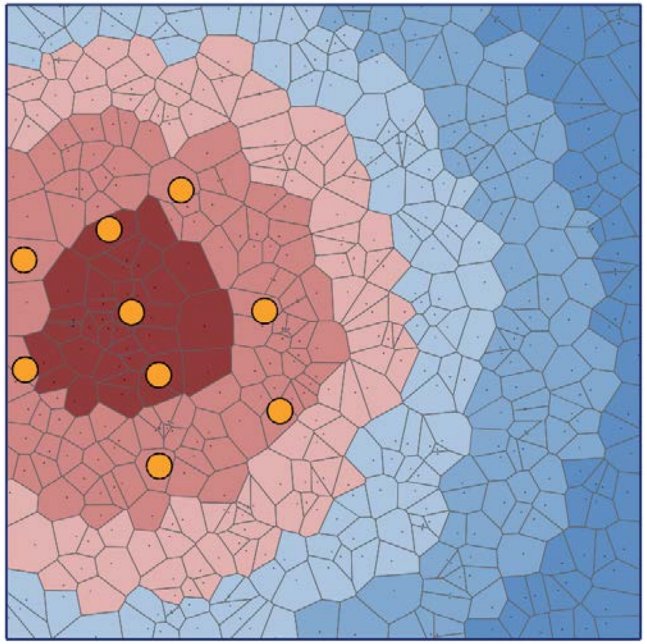
- Objective Function:

$$\min 1^T x + \lambda x^T C x$$

Fit and Diverse Sampling

Current Work

- Input: set of samples in a domain
  f - fitness scores for each sample
  S - similarity score matrix
  k – number of samples to be selected
- Goal: select a set of samples that is fit and diverse



- Variables: $x_i = 1$ if sample i is selected
- Sum of selected samples constraint:

$$1^T x = k$$

- Objective function:

$$\max\ f^T x - x^T S x$$

(Mixed-)Integer Quadratic Programming with Quadratic Constraints

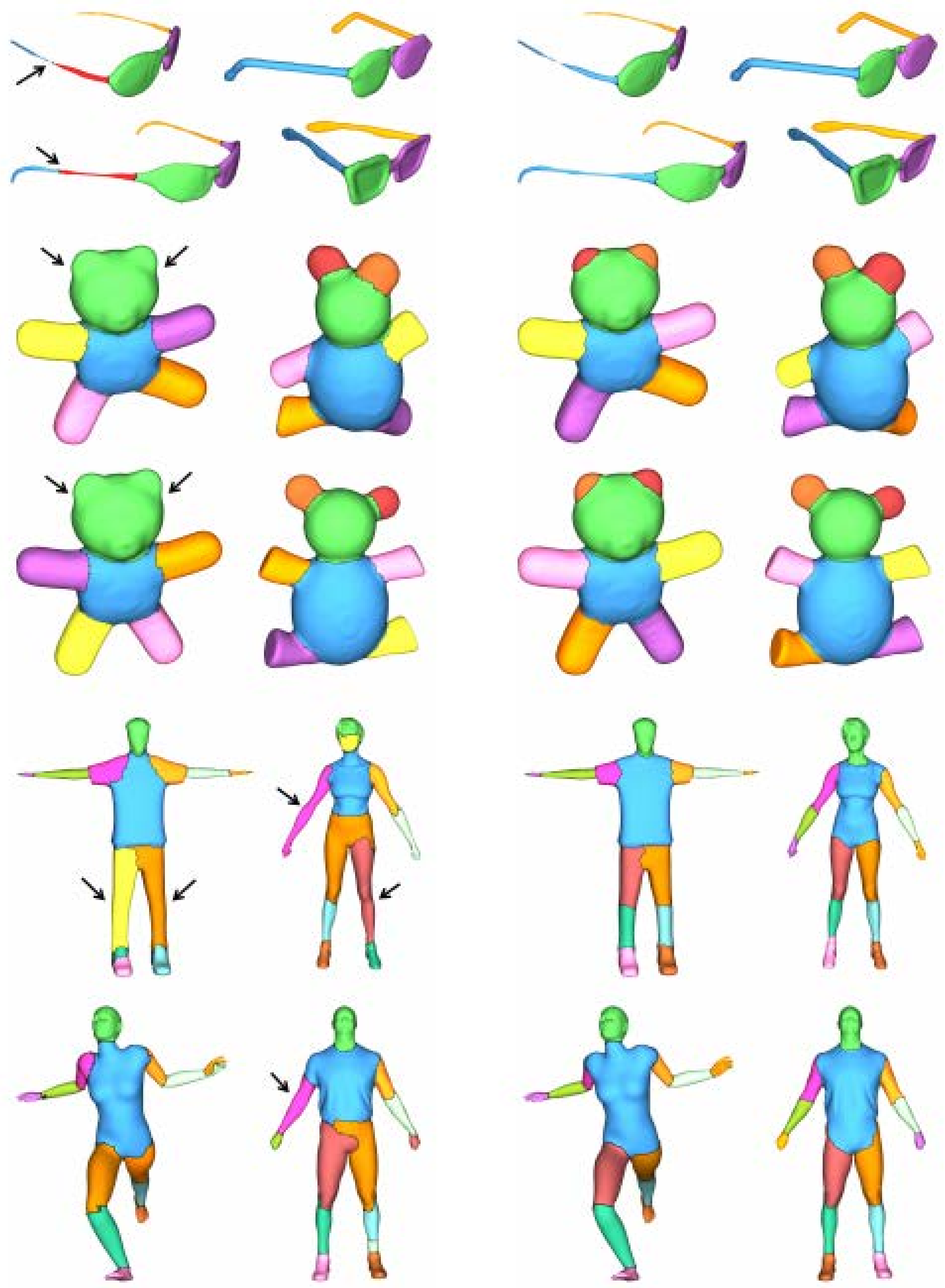
- General Form

$$\max cx + x^T C x$$

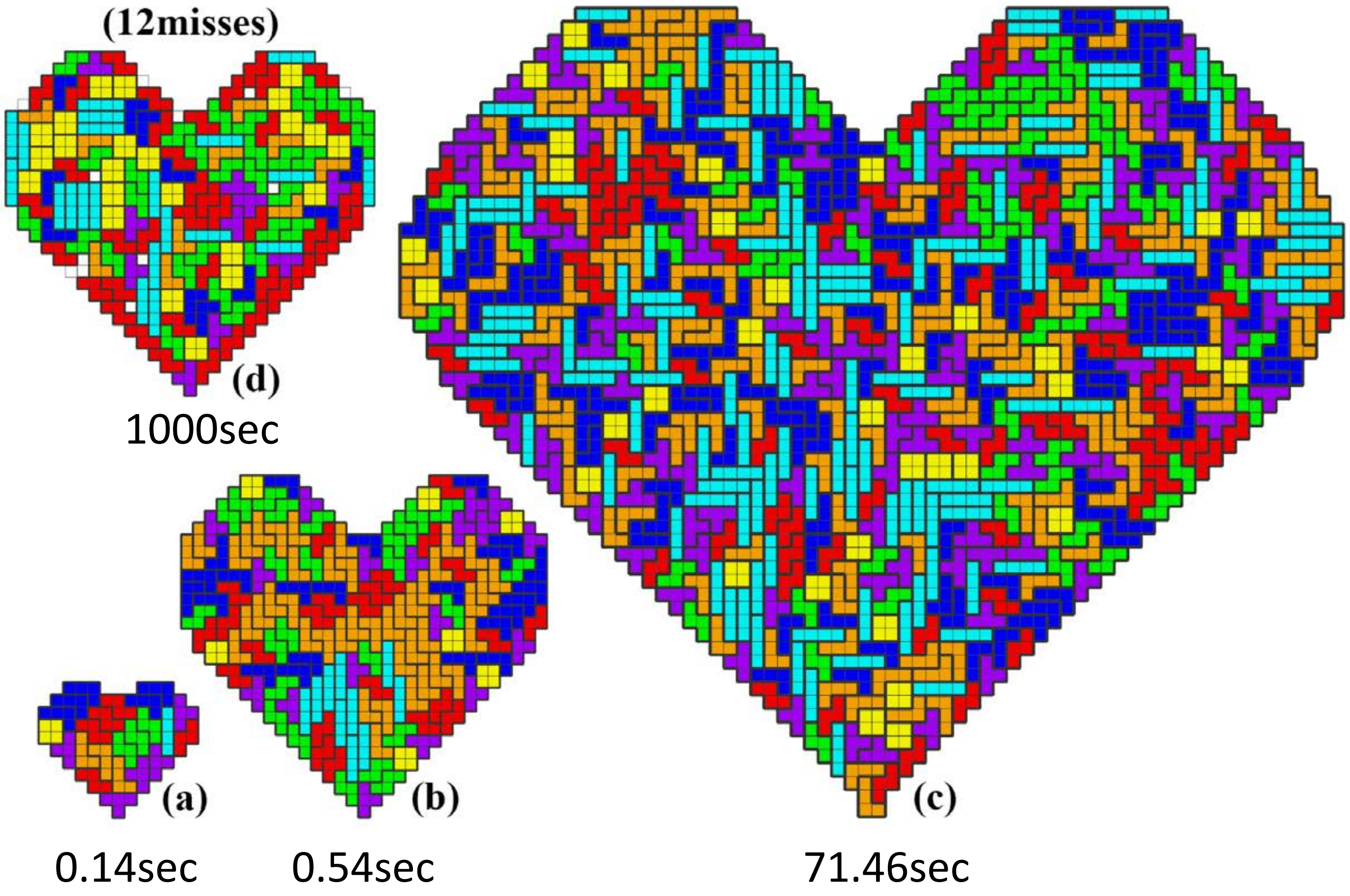
$$Ax \leq b$$

Joint Segmentation

SGA 2011, Huang et al.



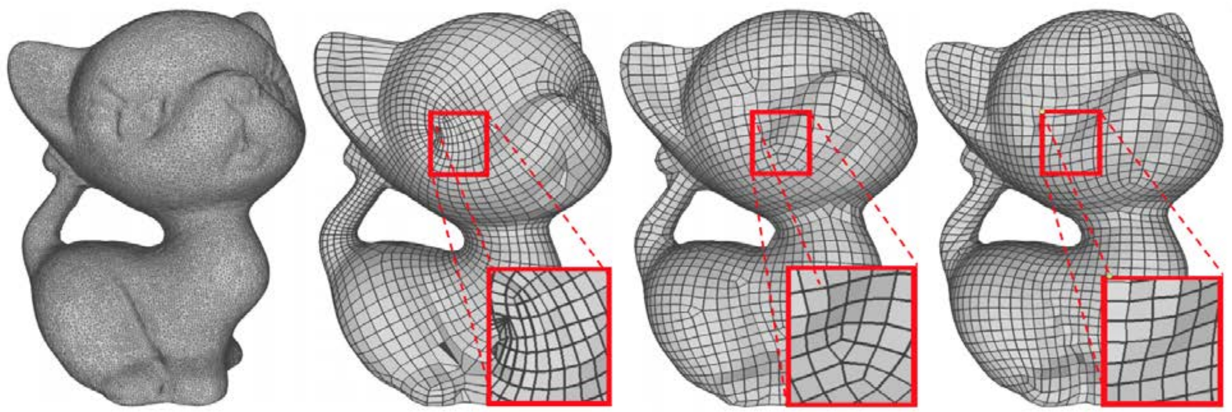
Performance considerations of IP



[SG 2014]

(a), (b), and (c):
solved by a general-purpose solver (Gurobi)

(d):
solved by a greed method with simulated annealing



(a) Input triangulation (b) Integer-Grid Maps (c) Instant Meshes (d) QuadriFlow

$$
\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & x_1 + 2x_2 + 3x_3 + x_4 + 4x_5 \\
\text{subject to} \quad & x_2 - x_3 = -5 \\
& x_5 - x_2 - x_1 = -2 \\
& x_3 + x_4 - x_5 = 4 \\
& x_1 - x_4 = 3 \\
& 0 \le x_1 \le 3 \\
& 0 \le x_2 \le 2 \\
& 0 \le x_3 \le 2 \\
& 0 \le x_4 \le 3 \\
& 0 \le x_5 \le 1 \\
& x_i \in \mathbb{Z} \quad \forall i
\end{aligned}
$$

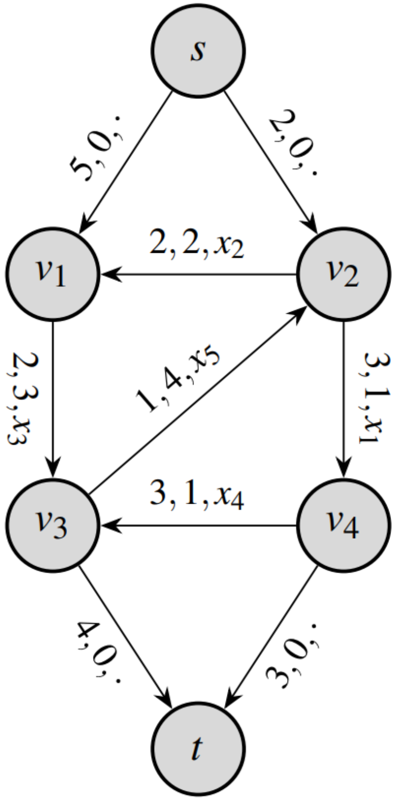


Table 4: *Comparison of multiple methods for integer optimization. We show the running times, average angle distortions, and average area distortions on two test examples. The number 900 or 1,500 represents the specified edge density. MF, MCF, MR, and ILP stand for maximum flow, minimum cost flow, multi-resolution, and integer linear programming, respectively.*

Mesh & Algorithm	Time	Angle error	Area error
Hand_900_MF	0.85	**11.195277**	0.272820
Hand_900_MF_MR	**0.09**	12.695140	**0.237884**
Hand_900_MCF	4.12	12.555485	0.294125
Hand_900_MCF_MR	0.11	13.011465	0.263241
Hand_900_ILP	280.00	12.555485	0.294125
Hand_1500_MF	1.76	9.387929	**0.193454**
Hand_1500_MF_MR	**1.05**	10.391423	0.205469
Hand_1500_MCF	13.41	**8.786778**	0.210081
Hand_1500_MCF_MR	1.09	8.982389	0.220997
Hand_1500_ILP	164.00	**8.786778**	0.210081

QuadriFlow: A Scalable and Robust Method for Quadrangulation. Jingwei Huang et al, SGP 2018

Thank You!