ET4255 Programming Assignment

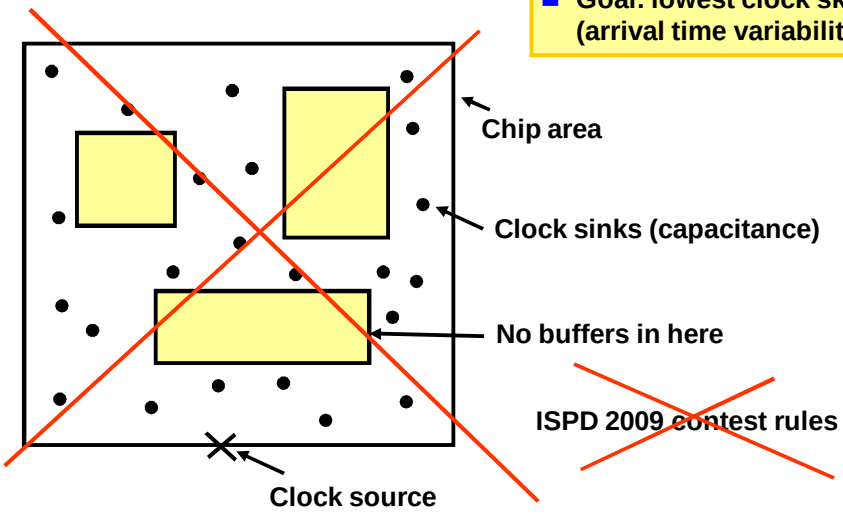
- Clock network synthesis from ISPD 2009 contest
- Solve a (simplified derivative of) CAD problem
- Prize and eternal fame for best solutions

- May program in any language you like, feel comfortable with, any platform.
- Recommended: C++ on Linux (or cygwin)
- You may use the MSc lab machines on 16th floor
  - Contact me for how to obtain a login

- Clock network synthesis, simpler than ISPD 2009

Problem Structure

- Goal: lowest clock skew (arrival time variability)



Simplified Problem

- Path-based delay model
  (length of path is what matters, delay would be harder to compute and optimize)
- No buffering
- No blockages

- Starting Point:
  High Performance Clock Routing Based on Recursive Geometric Matching,
  Kahng, Cong and Robins,
  1991 Design Automation Conference

Problem Description

- **Path Balanced Tree Problem** [Kahng, Cong, Robins, 1991]:

  Given a set of points P in the L_1 unit square and a real number S, find a minimum cost tree T connecting P such that for some distinguished node r of T the costs of paths in T from r to any two leaf nodes differ by at most S.
- Actually: such that the cost difference is made as small as possible

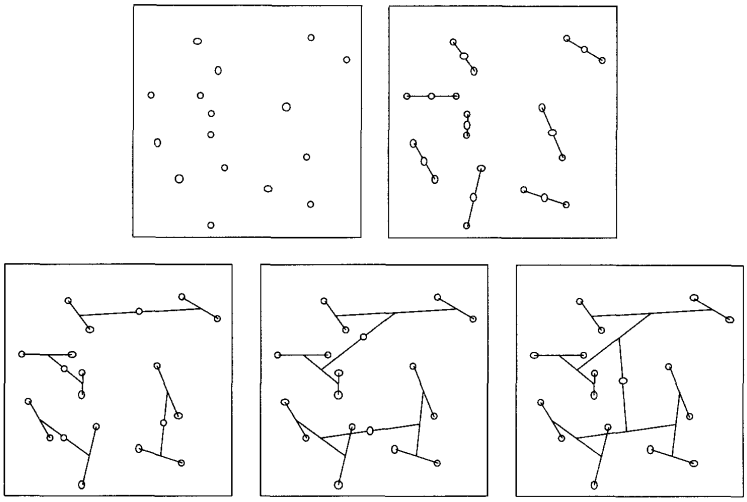
- L_1 unit square: Manhattan distance

- L_1 distance metric: $D_{L1}(p, q) = |p_x - q_x| + |p_y - q_y|$
  [L_n distance metric: $D_{Ln}(p, q) = ((p_x - q_x)^n + (p_y - q_y)^n)^{1/n}$]

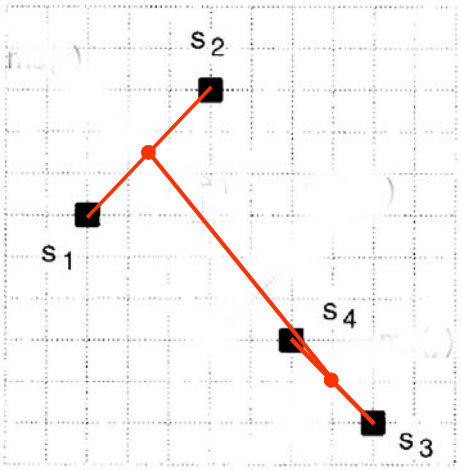
KCR: Abstract Topology Generation

Bottom-up point matching in the plane

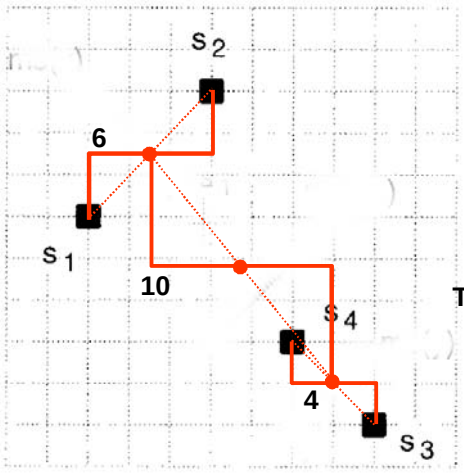
Forests of trees to be merged recursively



KCR Abstract Topology



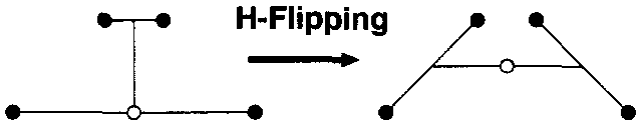
KCR Trivial Planar Manhattan Embedding



Total L_1 cost = 20
skew = 0

KCR H-Flipping

KCR does not guarantee zero-skew



no zero-skew **zero-skew**

Cost Function

- Total length of clock tree
- Absorb any remaining skew by detour

Non-zero skew

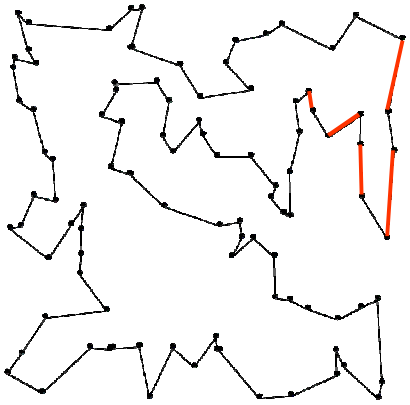
Skew Absorbed using Detour

KCR Algorithm

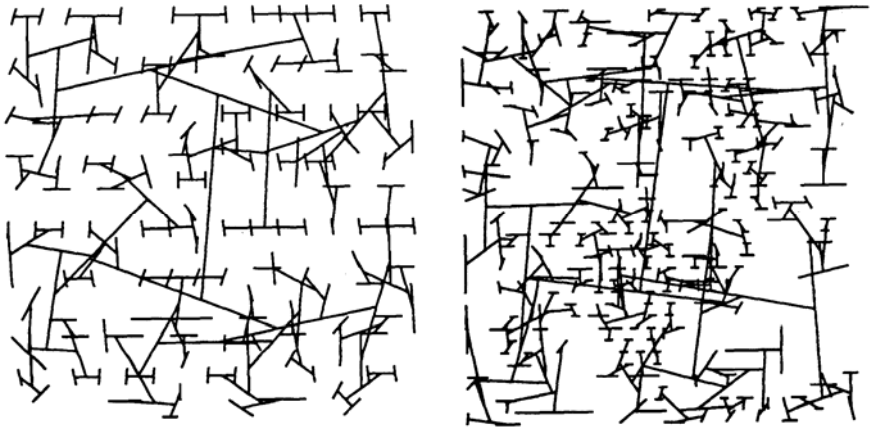
$T = \emptyset$
while $|P| > 1$
  M = optimal geometric matching over P
  $P' = \emptyset$
  for $(p_1, p_2) \in M$ **do**
    T_1 = subtree of T rooted at p_1
    T_2 = subtree of T rooted at p_2
    p = point on segment *between* p_1 and p_2
      such that p minimizes skew of tree
      $T_1 \cup T_2 \cup \{(p, p_1), (p, p_2)\}$ w/root p
    $P' = P' \cup \{p\}$
    $T = T \cup \{(p, p_1), (p, p_2)\}$
  $P = P'$ plus an unmatched node if $|P|$ odd
CEP = Root of T = single remaining point in P

Geometric Matching

- Optimal solution in $O(n^3)$ by Edmond's algorithm
- Many fast heuristics
- Heuristic for Traveling Salesman can be used
- But this is your play ground, suggestions in KCR paper



Select every other segment

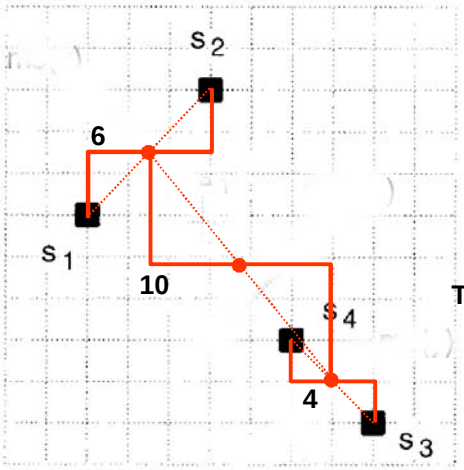


- KCR result on 2 benchmarks

Improvement

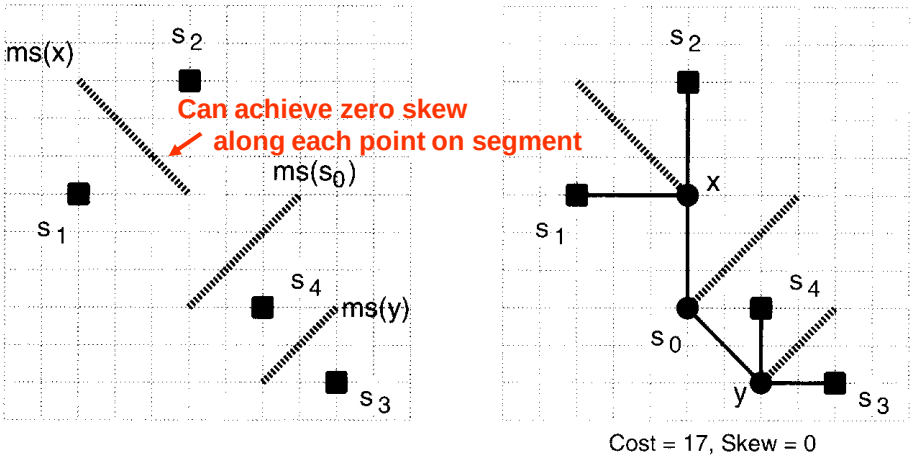
- Instead of trivial embedding, optimal zero skew embedding can be constructed
- Three independent discoveries within one year after KCR ☺
- KCR algorithm is mandatory – optimal embedding is optional
- See (e.g.)
  Performance optimization of VLSI interconnect layout,
  Jason Cong, Lei He, Cheng-Kok Koh, Patrick H. Madden,
  Integration, the VLSI Journal, 1996, p. 1-94

KCR Trivial Planar Manhattan Embedding



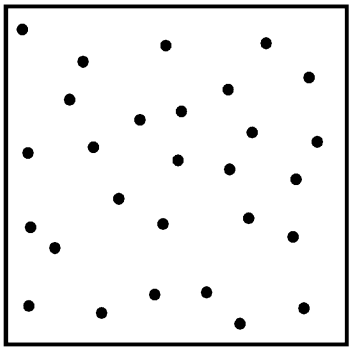
Total L_1 cost = 20
skew = 0

Deferred Merge Embedding (DME)



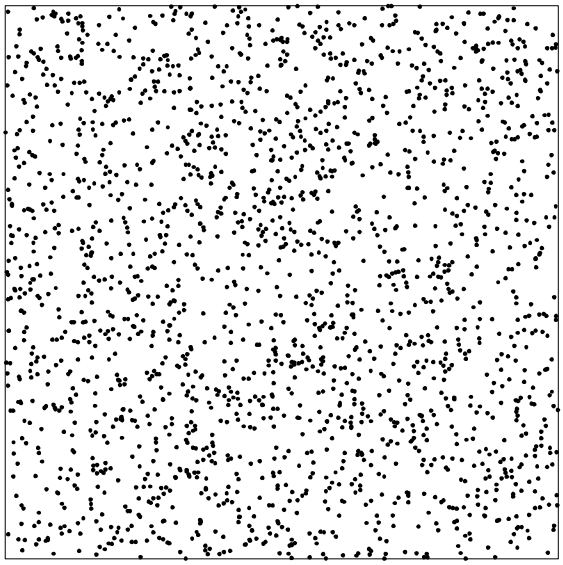
(b) Bottom-Up Merging Phase (c) Top-Down Embedding Phase

Logistics



- Input: such as figure on the left, using file with x,y coordinates
- Number of points is power of 2, and >> 1000
- Output: similar file
- To be made available: testing program that evaluates output file, measures path length and path length skew
- Also: test data sets
- Not: final test data
- Delivery: Mon, week 4 of Q2 (Nov 29).
- Work alone or with 2
- Everyone should do KCR, DME or other extension is optional
- Degrading policy for excessive runtime

Input Format

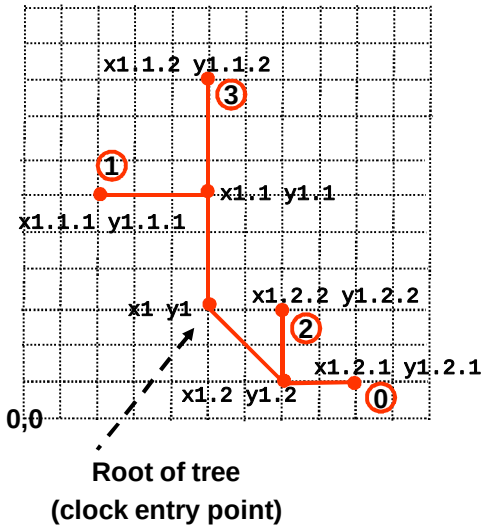


Example input file:

```
2048
1.2211351289 6.6476490799
7.8116659856 4.1836772553
3.7750290166 1.9158501140
1.4097627399 0.7319560371
6.2200532883 9.6860467362
4.9261750537 7.0067519820
1.4841317485 9.4780941212
2.2745055483 8.6676481732
8.1928937268 0.2288990003
Etc.
```

n = 2048

Output Format



Root of tree
(clock entry point)

```
num-of-sinks
x1      y1      0
x1.1    y1.1    d
x1.2    y1.2    d
x1.1.1  y1.1.1  d  id1
x1.1.2  y1.1.2  d  id2
x1.2.1  y1.2.1  d  id3
x1.2.2  y1.2.2  d  id4
```

template (definition)

d = detour

nodes ordered by breadth-first way

```
4
9 1
2 6
7 3
5 9
```

```
4
5 3 0
5 6 0
7 1 0
2 6 0 1
5 9 0 3
9 1 0 0
7 3 0 2
```

example input & output

Code: mk-input.c

```
#include <stdio.h>
#include <stdlib.h>
main (argc, argv)
int argc;
char * argv[];
{
    double x, y;
    int i, n;

    n = atoi (argv[1]);
    printf ("%d\n", n);

    // Use n as the seed:
    // repeatable but with different initial sequences
    // Will use other seed (or generator) 'for real'.
    srand (n);

    for (i = 0; i < n; i++) {
        x = ((double) rand ())/RAND_MAX;
        y = ((double) rand ())/RAND_MAX;
        printf ("%.10f %.10f\n", x*10.0, y*10.0);
    }

    exit (0);
}
```


Graphics

- Graphics format: SVG – Scalable Vector Graphics
- Can be rendered in some browsers (but still some bugs)
- INKSCAPE is freely available SVG drawing/rendering tool
- http://inkscape.org/

Code: plot-input.c

```
#include <stdlib.h>
#include <stdio.h>
#include "types.h"
int main (void)
{
    char buf[1000];
    point p;

    gets (buf); // gobble first line with count of points

    // print xml preamble
    printf ("<?xml version=\"1.0\" standalone=\"no\"?>\n");
    printf ("<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.1//EN\" ");
    printf ("\"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd\">\n");

    // setup canvas
    printf ("<svg width=\"1024\" height=\"1024\" version=\"1.1\" ");
    printf ("xmlns=\"http://www.w3.org/2000/svg\">\n");
    printf ("<rect x=\"12\" y=\"12\" width=\"1000\" height=\"1000\" ");
    printf ("style=\"fill:white; stroke-width:1; stroke:black\"/>\n");

    while (scanf ("%lf %lf", &p.x, &p.y) == 2) {
        if (p.x < 0 || p.y < 0 || p.x >= 10 || p.y >= 10) {
            fputs ("Input data out of bounds\n", stderr);
            return EXIT_FAILURE;
        }
        printf ("<circle cx=\"%f\" cy=\"%f\"",12+100 * p.x, 1012 - 100 * p.y);
        printf (" r=\"4\" stroke-width=\"0\" fill=\"black\"/>\n");
    }
    puts ("\n</svg>");
}
```


Deadline and Overall Rules

- **Deadline for delivery of code and documentation (electronically) is Monday Nov 29**
- **The code (and the documentation) should be yours. You should comply with all principles of Academic Honesty. Failure may lead to serious consequences.**
- **You are not allowed to use external libraries, except those that are standard with the programming language.**
- **However, you can use non-trivial external libraries if you have specific permission by me. You can get such permission e.g. if you intend to exceed the minimum requirements of the project.**
- **You can work in a team of max 2 students, but you should explain to me the breakdown of tasks and still you should both understand and be able to explain your own contribution as well as your partner's contribution**

Overall Grading Policy

- Overall grade determined by oral exam and project grade.
- Project includes code and documentation
- Weighting: 50% oral, 40% code, 10% documentation

- All partial scores should be at least 6.0
  Final score will be lowest of partial scores if one or more of them is lower then 6.0
- Penalty for late submission of project deliverables
  of 0.1 point/day on overall grade
  Max 1 point penalty

- A result can only count as correct when all remaining skew is subsumed in detours. Detour length is included in total path length L.
- Trivial results w/o any serious optimization of L don't count

- 1 point of code grade depends on solution quality, i.e. on path length L

Code Grading Policy

- 6 if acceptable, correct and non-trivial solution
- 7 if well-thought solution, showing understanding and extra mileage in quality of test data, robustness, efficiency
- 1 extra point for correct DME extension
- Max 1 extra point for code quality
  (readability, simplicity, maintainability, clarity, internal documentation)
- $0 \le x \le 1$ point for solution quality, based on rank as determined by total path length L.
  $(x = (N - \text{rank})/(N-1))$ N is number of participating teams
- Max 1 extra point for original contribution beyond state of art or useful extension of assignment (blockages?)
- Code grade is bounded by 10

Documentation Grading Policy

- Document should document overall flow of code, and clearly indicate and motivate major design designs. Should help reader in understanding the code.
- Should also include correct references
- Don't forget title page including your name(s) and reference to course/assignment
- It need not be long
  (2 pages using normal page layout and font size can suffice)
- Minimum 6 for bare document
- Max 1 point for helpful diagrams/figures
- Max 2 point for organization, neatness, readability, clarity, style, language, …
- Max 1 point for closing in on publication quality

Questions?