Optimisation Applications

Leandro L. Minku

Overview

- Reiterate some concepts and ideas.
- Briefly mention some applications of Hill Climbing and Simulated Annealing.
- One application of Simulated Annealing in detail.

Hill Climbing Applications

- Hill-Climbing is (arguably) applicable to any optimisation problem.
- Simple algorithm — not difficult to implement.
  - Could be attempted first to see if the retrieved solutions are good enough, before a more complex algorithm is investigated.
  - Its success depends on the shape of the objective function for the problem instance in hands.
  - The need for defining a neighbourhood operator can make it more suitable for discrete optimisation problems.

Applications

- Hill-climbing has been successfully applied to software module clustering.

- Software Module Clustering:
  - Software is composed of several units, which can be organised into modules.
  - Well modularised software is easier to develop and maintain.
  - As software evolves, modularisation tends to degrade.

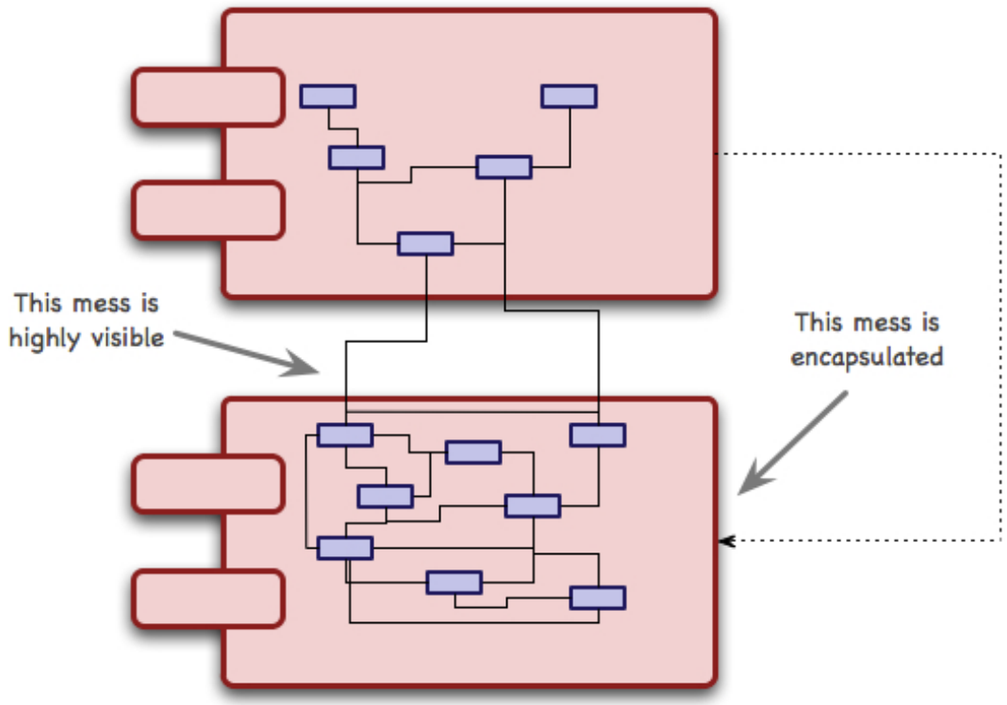


Image from: http://www.kirkk.com/modularity/wp-content/uploads/2009/12/EncapsulatingDesign1.jpg

Problem: find an allocation of units into modules that maximises the quality of modularisation.

Simulated Annealing Applications

- Simulated Annealing is (arguably) also applicable to any optimisation problem.
- The need for defining a neighbourhood operator can make it more suitable for discrete optimisation problems.
- Some problems may be easier or more difficult for Simulated Annealing to solve well, and it may not be possible to know beforehand how well Simulated Annealing will work.
  - This is an issue with many of the Artificial Intelligence algorithms.
- One possibility is to check whether other similar problems have been well solved well by Simulated Annealing before.
  - This can be helpful, though it does not guarantee that Simulated Annealing is a good algorithm for the problem in hands.

Examples of Applications

- Software engineering problems:
  - Component selection and prioritisation for the next release problem.
  - Software quality prediction.

Wind Farm Optimisation

[Video posted by the Optimisation and Logistics group from the University of Adelaide: http://cs.adelaide.edu.au/~optlog/research/energy.php]

Examples of Applications of Simulated Annealing

- Several engineering problems, e.g.: VLSI (Very-Large-Scale Integration).
  - Process of creating an integrated circuit by combining thousands of transistors into a single chip.
  - Decide placement of transistors.
  - Objectives: minimise area, wiring and congestion.
  - Constraints: ensuring that the transistors are placed within a certain maximum area and are not in clashing positions.

Image from: https://upload.wikimedia.org/wikipedia/commons/9/94/VLSI_Chip.jpg

Applying Hill-Climbing or Simulated Annealing

- We need to specify:
  - Optimisation problem formulation:
    - Design variable and search space
    - Constraints
    - Objective function
  - Algorithm-specific operators:
    - Representation.
    - Initialisation procedure.
    - Neighbourhood operator.
  - Strategy to deal with constraints, e.g.:
    - Representation, initialisation and neighbourhood operators that ensure only feasible solutions to be generated.
    - Modification in the objective function.

Applying Hill-Climbing or Simulated Annealing

- We need to specify:
  - **Optimisation problem formulation:**
    - Design variable and search space
    - Constraints
    - Objective function
  - Algorithm-specific operators:
    - Representation.
    - Initialisation procedure.
    - Neighbourhood operator.
  - Strategy to deal with constraints, e.g.:
    - Representation, initialisation and neighbourhood operators that ensure only feasible solutions to be generated.
    - Modification in the objective function.

Multi-Objective Optimisation Problems

Objective functions

Design variables

Minimise $f_k(\mathbf{x}), \quad k = 1 \cdots, p$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

Constraints

Search space: space of all possible $\mathbf{x}$ values.

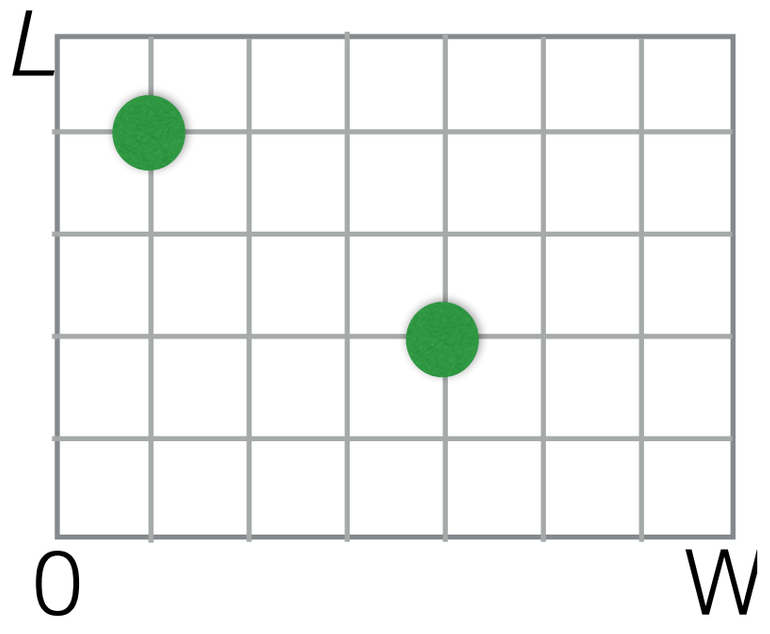
Optimising VLSI Transistor Placement

- Design variable:

  candidate solution $\mathbf{z} = [(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$,

  where N is the number of transistors, $\forall i \in \{1,2,\ldots,N\}$, $x_i \in Z$ and $y_i \in Z$ are the coordinates of transistor i.

We want to decide where to place each of N transistors so as to minimise area of the chip, length of wiring and congestion.



Optimising VLSI Transistor Placement

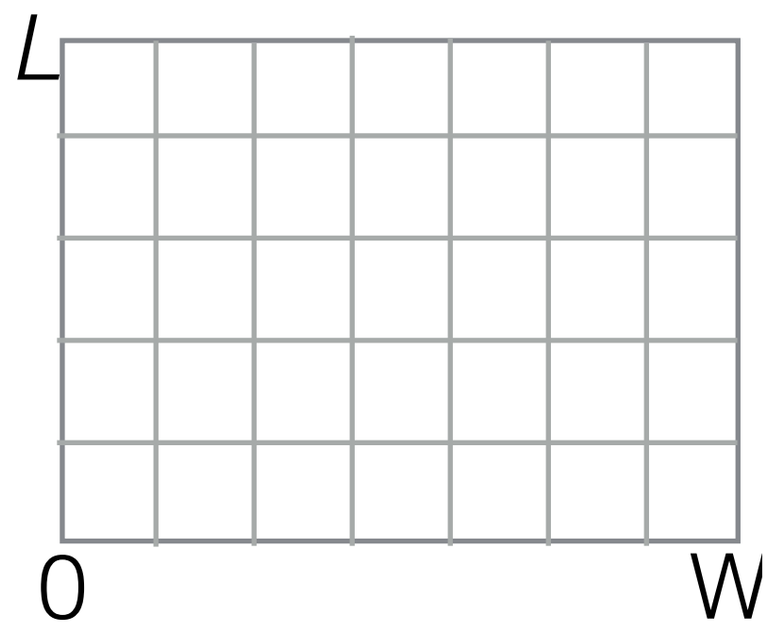
- Design variable:

  candidate solution $\mathbf{z} = [(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$,

  where N is the number of transistors, $\forall i \in \{1,2,\ldots,N\}$, $x_i \in Z$ and $y_i \in Z$ are the coordinates of transistor i.

- Objectives: minimise $f_1(\mathbf{z})$ = area of the chip, $f_2(\mathbf{z})$ = length of wiring and $f_3(\mathbf{z})$ = congestion.

How to deal with multiple objectives?



Optimising VLSI Transistor Placement

- Design variable:

  candidate solution $\mathbf{z} = [(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$,

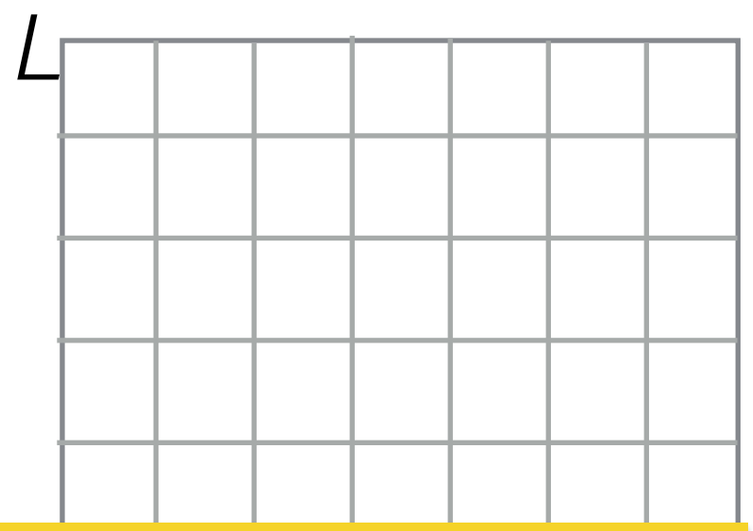
  where N is the number of transistors, $\forall i \in \{1,2,\ldots,N\}$, $x_i \in Z$ and $y_i \in Z$ are the coordinates of transistor i.

- Objectives: minimise $f_1(\mathbf{z})$ = area of the chip, $f_2(\mathbf{z})$ = length of wiring and $f_3(\mathbf{z})$ = congestion.

  Minimise $f(\mathbf{z}) = a\ f_1(\mathbf{z}) + b\ f_2(\mathbf{z}) + c\ f_3(\mathbf{z})$

  where $a,b,c \in [0,1]$ and

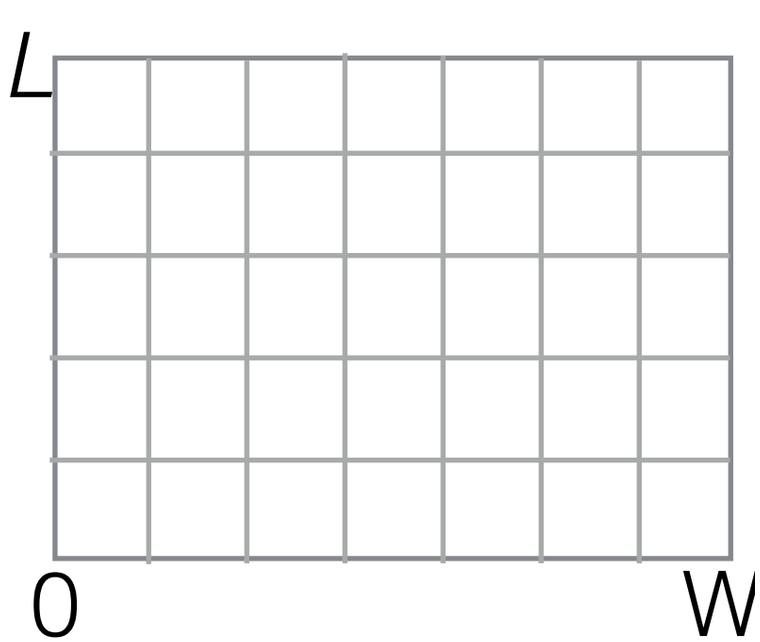
  $a + b + c = 1$.



There are other strategies to deal with multiple objectives, including algorithms specifically designed for multiple objectives.

Optimising VLSI Transistor Placement

- Constraints:
  - For $\forall i \in \{1,\ldots,N\}$:
    - $x_i \geq 0$
    - $x_i \leq W$
    - $y_i \geq 0$
    - $y_i \leq L$
    - For $\forall j \in \{i+1,\ldots,N\}$:
      - $(x_i, y_i) \neq (x_j, y_j)$



How to convert to the canonical format?

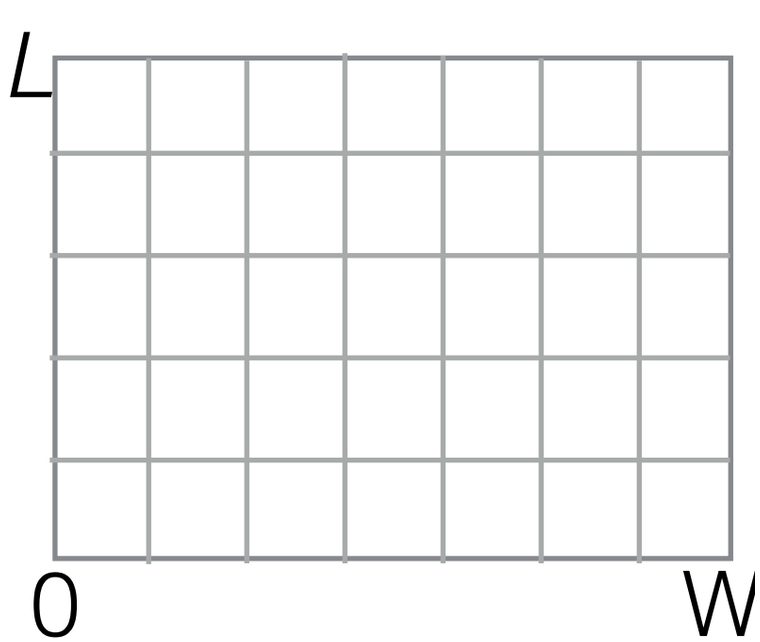
Optimising VLSI Transistor Placement

- Constraints:
  - For ∀i ∈ {1,…,N}:
    - $x_i \geq 0$ $-x_i \leq 0$
    - $x_i \leq W$ $x_i - W \leq 0$
    - $y_i \geq 0$ $-y_i \leq 0$
    - $y_i \leq L$ $y_i - L \leq 0$
  - For ∀j ∈ {i+1,…,N}:
    - $(x_i, y_i) \neq (x_j, y_j)$

$|x_i - x_j| + |y_i - y_j| \neq 0$ $|x_i - x_j| + |y_i - y_j| > 0$

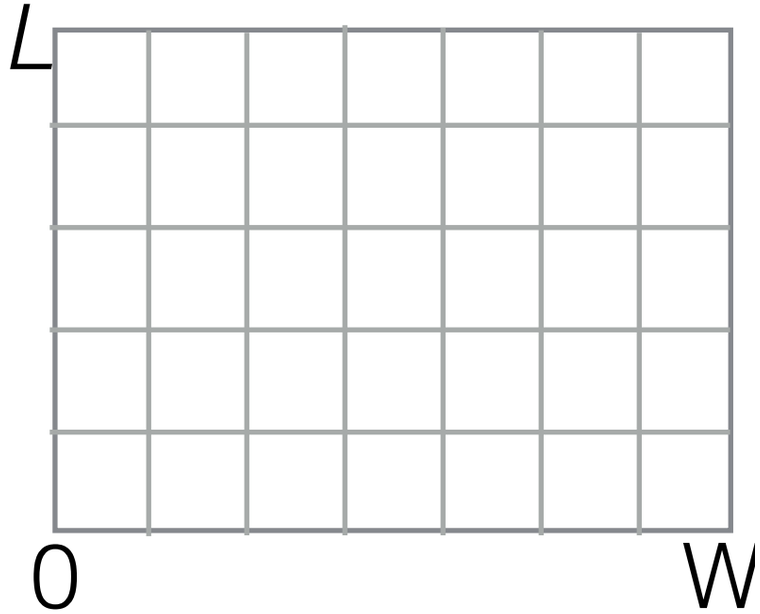
$|x_i - x_j| + |y_i - y_j| \geq 1$ $|x_i - x_j| + |y_i - y_j| - 1 \geq 0$

$-|x_i - x_j| - |y_i - y_j| + 1 \leq 0$



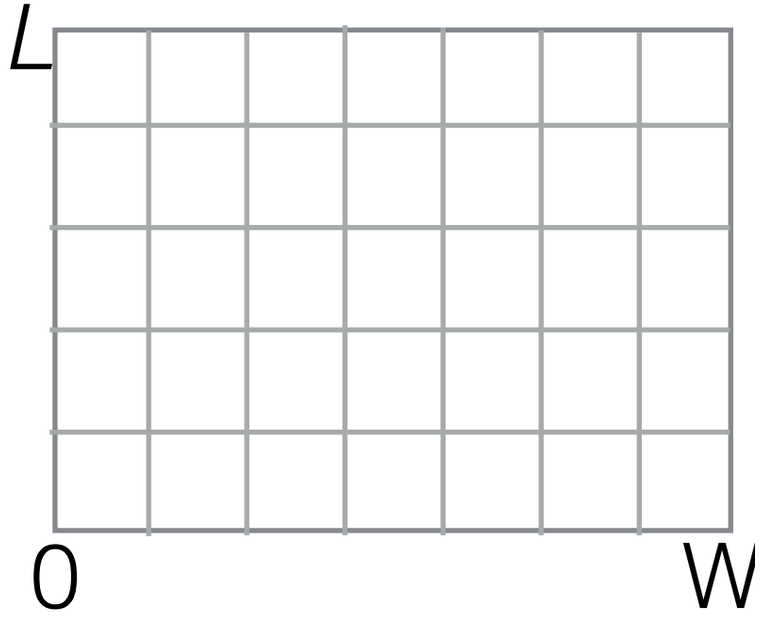
Optimising VLSI Transistor Placement

- Constraints:
  - For $\forall i \in \{1,\ldots,N\}$:
    - $-x_i \leq 0$
    - $x_i - W \leq 0$
    - $-y_i \leq 0$
    - $y_i - L \leq 0$
    - For $\forall j \in \{i+1,\ldots,N\}$:
      - $-|x_i - x_j| - |y_i - y_j| + 1 \leq 0$



Optimising VLSI Transistor Placement

- Constraints:
  - For $\forall i \in \{1,\ldots,N\}$:
    - $g_{(i-1)*4+1}(\mathbf{z}) = -x_i \leq 0$
    - $g_{(i-1)*4+2}(\mathbf{z}) = x_i - W \leq 0$
    - $g_{(i-1)*4+3}(\mathbf{z}) = -y_i \leq 0$
    - $g_{(i-1)*4+4}(\mathbf{z}) = y_i - L \leq 0$
  - For $\forall j \in \{i+1,\ldots,N\}$:
    - $g_{ij}(\mathbf{z}) = -|x_i - x_j| - |y_i - y_j| + 1 \leq 0$



Applying Simulated Annealing (and Hill-Climbing)

- We need to specify:
  - Optimisation problem formulation:
    - Design variable and search space
    - Constraints
    - Objective function
  - **Algorithm-specific operators:**
    - Representation.
    - Initialisation procedure.
    - Neighbourhood operator.
  - **Strategy to deal with constraints, e.g.:**
    - Representation, initialisation and neighbourhood operators that ensure only feasible solutions to be generated.
    - Modification in the objective function.

Applying Simulated Annealing (and Hill-Climbing)

- We need to specify:
  - Optimisation problem formulation:
    - Design variable and search space
    - Constraints
    - Objective function
  - **Algorithm-specific operators:**
    - **Representation.**
    - **Initialisation procedure.**
    - **Neighbourhood operator.**
  - **Strategy to deal with constraints, e.g.:**
    - **Representation, initialisation and neighbourhood operators that ensure only feasible solutions to be generated.**
    - Modification in the objective function.

—> for dealing with the within bound constraints

Representation and Initialisation

- Representation: direct representation of the design variable

  candidate solution is a vector of size N of 2-d vectors
  $\mathbf{z} = [(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$,

  where N is the number of transistors,
  $x_i \in Z$ and $y_i \in Z$, $\forall i \in \{1,2,\ldots,N\}$ are the coordinates of transistor i.

- Initialisation:
  - $\forall i \in \{1,2,\ldots,N\}$, initialise x_i, uniformly at random with integer values in $0 \leq x_i \leq W$.
  - $\forall i \in \{1,2,\ldots,N\}$, initialise y_i, uniformly at random with integer values in $0 \leq y_i \leq L$.

Representation and Initialisation

- Neighbourhood operator for Simulated Annealing:
    1. Pick transistor i in {1,…,N} uniformly at random.
    2. Pick a coordinate x_i or y_i uniformly at random.
    3. Pick an operation +1 or -1 uniformly at random.
    4. Apply operation to the chosen coordinate of transistor i to generate a neighbour.
    5. If the neighbour's coordinate is $x_i < 0$
        - coordinate $x_i = 0$
    6. If the neighbour's coordinate is $x_i > W$
        - coordinate $x_i = W$
    7. If the neighbour's coordinate is $y_i < 0$
        - coordinate $y_i = 0$
    8. If the neighbour's coordinate is $y_i > L$
        - coordinate $y_i = L$

- PS: this operator may generate a neighbour that is a copy of the current solution. How to prevent that?

Applying Simulated Annealing (and Hill-Climbing)

- We need to specify:
  - Optimisation problem formulation:
    - Design variable and search space
    - Constraints
    - Objective function
  - Algorithm-specific operators:
    - Representation.
    - Initialisation procedure.
    - Neighbourhood operator.
  - **Strategy to deal with constraints, e.g.:**
    - Representation, initialisation and neighbourhood operators that ensure only feasible solutions to be generated.
    - **Modification in the objective function.**

—> for dealing with the transistor clash constraint

Using the Level of Infeasibility

Minimise $f(\mathbf{x}) + Q(\mathbf{x})$

Subject to $g_i(\mathbf{x}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{x}) = 0, \quad j = 1, \cdots, n$

$$Q(\mathbf{x}) = v_{g_1} C g_1(\mathbf{x})^2 + v_{g_2} C g_2(\mathbf{x})^2 + \ldots + v_{g_m} C g_m(\mathbf{x})^2 + v_{h_1} C h_1(\mathbf{x})^2 + v_{h_2} C h_2(\mathbf{x})^2 + \ldots + v_{h_n} C h_n(\mathbf{x})^2$$

$$Q(\mathbf{x}) = \sum_{i=1}^{m} v_{g_i} C g_i(\mathbf{x})^2 + \sum_{i=1}^{n} v_{h_i} C h_i(\mathbf{x})^2$$

Optimising VLSI Transistor Placement

- Design variable:

  candidate solution $\mathbf{z} = [(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$,

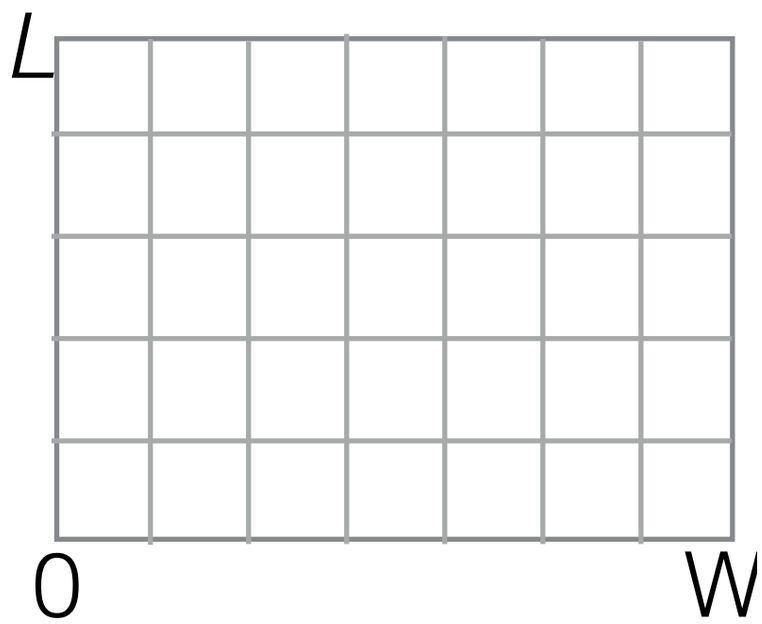
  where N is the number of transistors, $\forall i \in \{1,2,\ldots,N\}$, $x_i \in Z$ and $y_i \in Z$ are the coordinates of transistor i.

- Objectives: minimise $f_1(\mathbf{z})$ = area of the chip, $f_2(\mathbf{z})$ = length of wiring and $f_3(\mathbf{z})$ = congestion.

  Minimise $f(\mathbf{z}) = a\, f_1(\mathbf{z}) + b\, f_2(\mathbf{z}) + c\, f_3(\mathbf{z})$

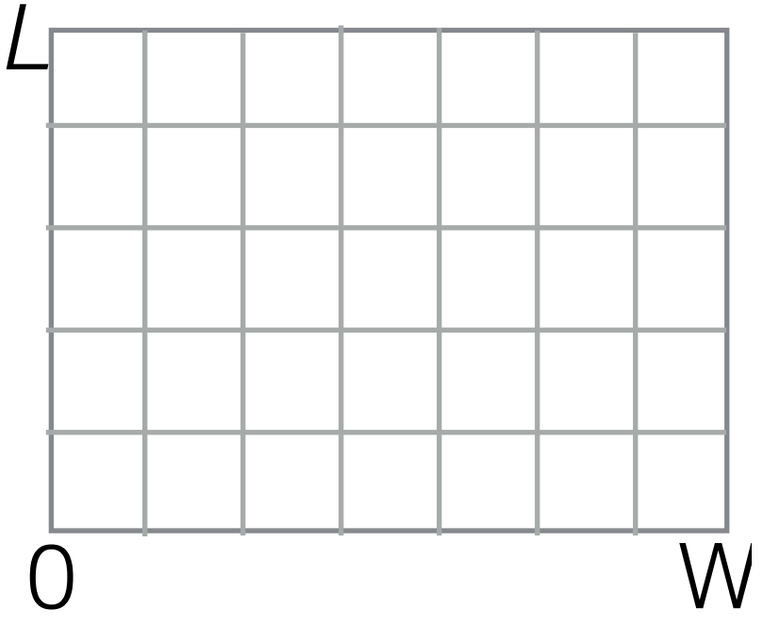
  where $a,b,c \in [0,1]$ and

  $a + b + c = 1$.



Optimising VLSI Transistor Placement

- Constraints:
  - For $\forall i \in \{1,\ldots,N\}$:
    - $g_{(i-1)*4+1}(\mathbf{z}) = -x_i \leq 0$
    - $g_{(i-1)*4+2}(\mathbf{z}) = x_i - W \leq 0$
    - $g_{(i-1)*4+3}(\mathbf{z}) = -y_i \leq 0$
    - $g_{(i-1)*4+4}(\mathbf{z}) = y_i - L \leq 0$
  - For $\forall j \in \{i+1,\ldots,N\}$:
    - $g_{ij}(\mathbf{z}) = -|x_i - x_j| - |y_i - y_j| + 1 \leq 0$



Using the Level of Infeasibility

Minimise $f(\mathbf{z}) + Q(\mathbf{z})$

Subject to $g_i(\mathbf{z}) \leq 0, \quad i = 1, \cdots, m$

$h_j(\mathbf{z}) = 0, \quad j = 1, \cdots, n$

$$Q(\mathbf{z}) = v_{g_1} C g_1(\mathbf{z})^2 + v_{g_2} C g_2(\mathbf{z})^2 + \ldots + v_{g_m} C g_m(\mathbf{z})^2 + v_{h_1} C h_1(\mathbf{z})^2 + v_{h_2} C h_2(\mathbf{z})^2 + \ldots + v_{h_n} C h_n(\mathbf{z})^2$$

$$Q(\mathbf{x}) = \sum_{i=1}^{m} v_{g_i} C g_i(\mathbf{x})^2 + \sum_{i=1}^{n} v_{h_i} C h_i(\mathbf{x})^2$$

Using the Level of Infeasibility

Minimise $f(\mathbf{z}) + Q(\mathbf{z})$

Subject to $g_i(\mathbf{z}) \leq 0, \quad i = 1, \cdots, m$

~~$h_j(\mathbf{z}) = 0, \quad j = 1, \cdots, n$~~

$$Q(\mathbf{z}) = v_{g_1} C g_1(\mathbf{z})^2 + v_{g_2} C g_2(\mathbf{z})^2 + \ldots + v_{g_m} C g_m(\mathbf{z})^2$$

~~$+ v_{h_1} C h_1(\mathbf{z})^2 + v_{h_2} C h_2(\mathbf{z})^2 + \ldots + v_{h_n} C h_n(\mathbf{z})^2$~~

$$Q(\mathbf{z}) = \sum_{i=1}^{m} v_{g_i} g_i(\mathbf{z})^2$$

Using the Level of Infeasibility

Constraints:

- For ∀i ∈ {1,…,N}:
  - ~~$g_{(i-1)*4+1}(\mathbf{z}) = -x_i \leq 0$~~
  - ~~$g_{(i-1)*4+2}(\mathbf{z}) = x_i - W \leq 0$~~
  - ~~$g_{(i-1)*4+3}(\mathbf{z}) = -y_i \leq 0$~~
  - ~~$g_{(i-1)*4+4}(\mathbf{z}) = y_i - L \leq 0$~~
  - For ∀j ∈ {i+1,…,N}:
    - $g_{ij}(\mathbf{z}) = -|x_i - x_j| - |y_i - y_j| + 1 \leq 0$

~~$Q(\mathbf{z}) = \sum_{i=1}^{m} v_{g_i} C g_i(\mathbf{z})^2$~~

$$Q(\mathbf{z}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} v_{g_{ij}} C g_{ij}(\mathbf{z})^2$$

where C is a large positive constant; and $\forall i, j \in \{1,...,N\}$, $v_{g_{ij}}$ is 1 if $g_{ij}(\mathbf{z})$ is violated and 0 otherwise.

Summary

- VLSI transistor placement problem formulation (except for objective function).
- Representation, neighbourhood operator, initialisation procedure.
- Strategy to deal with constraints.

Next

- Machine learning.