Optimization + Exploration

Jonathan Bachrach

EECS UC Berkeley

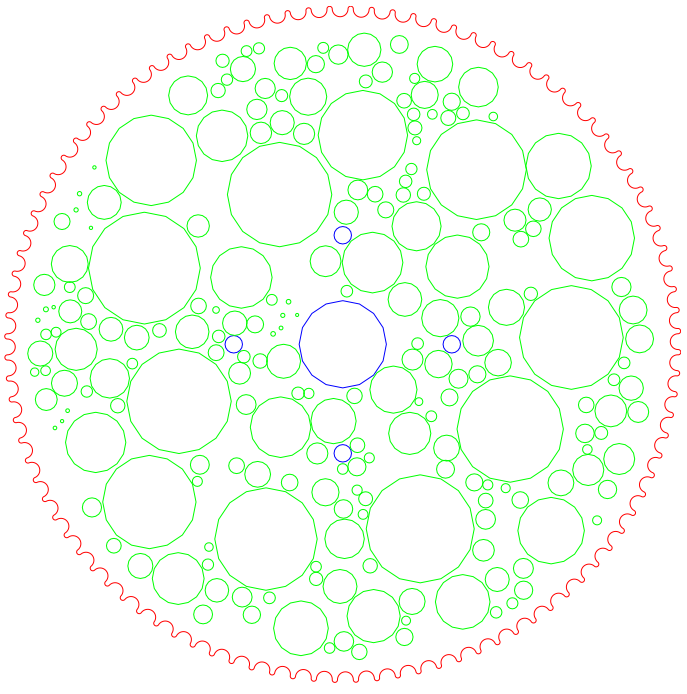
October 15, 2015

News

- lab 6 out later today
- one more week of lecture
- then project proposals
- next week generative design + future fab ...

Last Time: Declarative Design

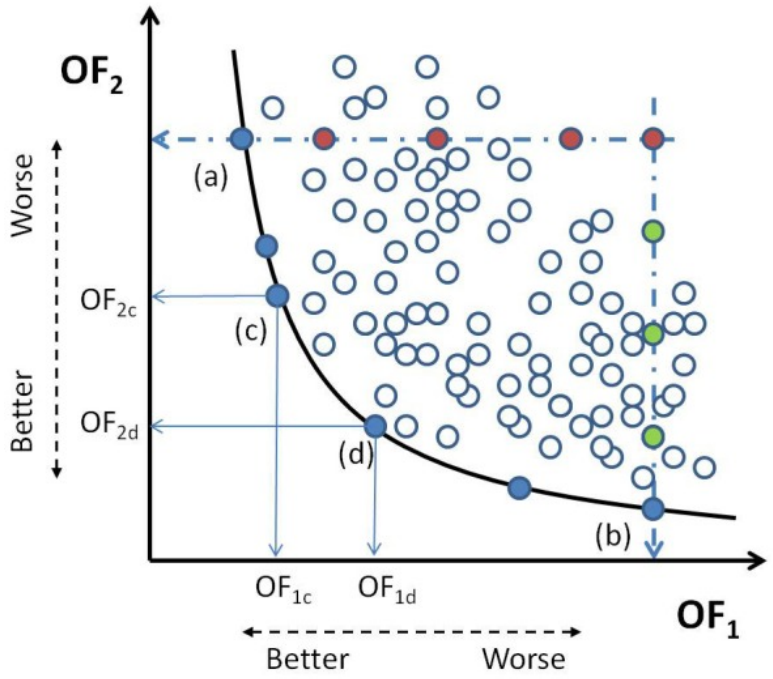
- What not How?
- Articulate Specification



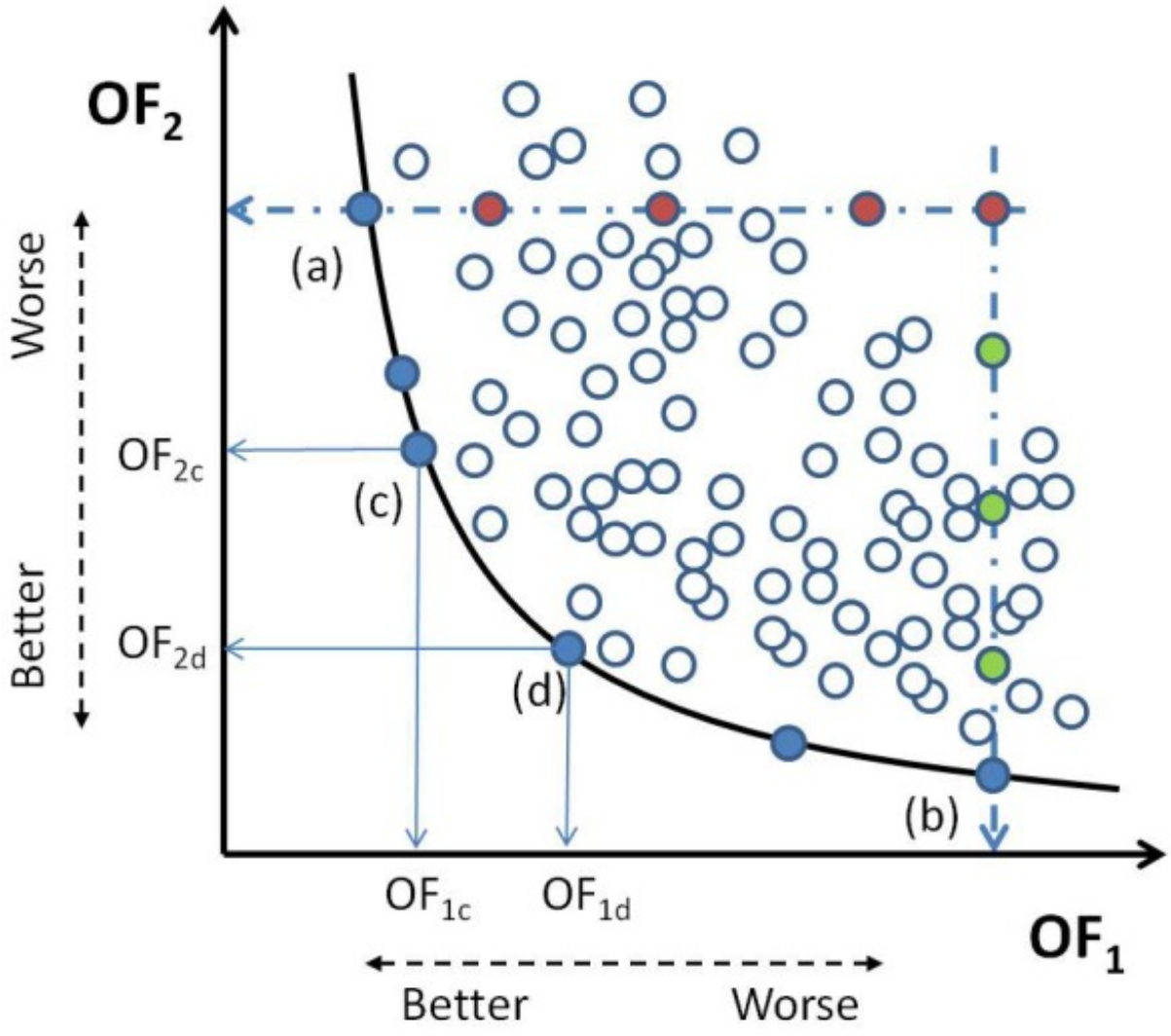
This Time: DSE + Optimization

Find the “best”

- Design Space
- Parameters
- Constraints
- Evaluation
- Optimization
- Exploration



Energy Delay Plot



Design Space Exploration

- complexity
  - solution space – (e.g., joules / op, area, ...)
  - problem space – (e.g., cache size, number cores, ...)
- optimization strategies
  - pareto optimality – dominates
  - blended solution – cost function with constraints
- evaluation
  - simulation
  - analytical
- exploration
  - basic
  - pruning

from Building ASIPS: The MESCAL Methodology by Gries, Keutzer, Meyr, Martin Springer Book

- primary
  - cost
  - power dissipation
  - speed
  - flexibility
- combined
  - energy-delay product
  - computations-power ratio
  - speed-cost ratio
  - flexibility-related

Problem Space – Parameters

- continuous
  - linear
  - log
- discrete
  - boolean
  - enum
- implementations
  - named choices
  - functional description that also take parameters

Pareto Dominance and Optimality

- assume n objectives
- pareto dominant solution
  - is $>$ in one objective while being $>=$ in others
- pareto optimal solution
  - if no other solution dominates
  - all elements in set are reasonable solutions

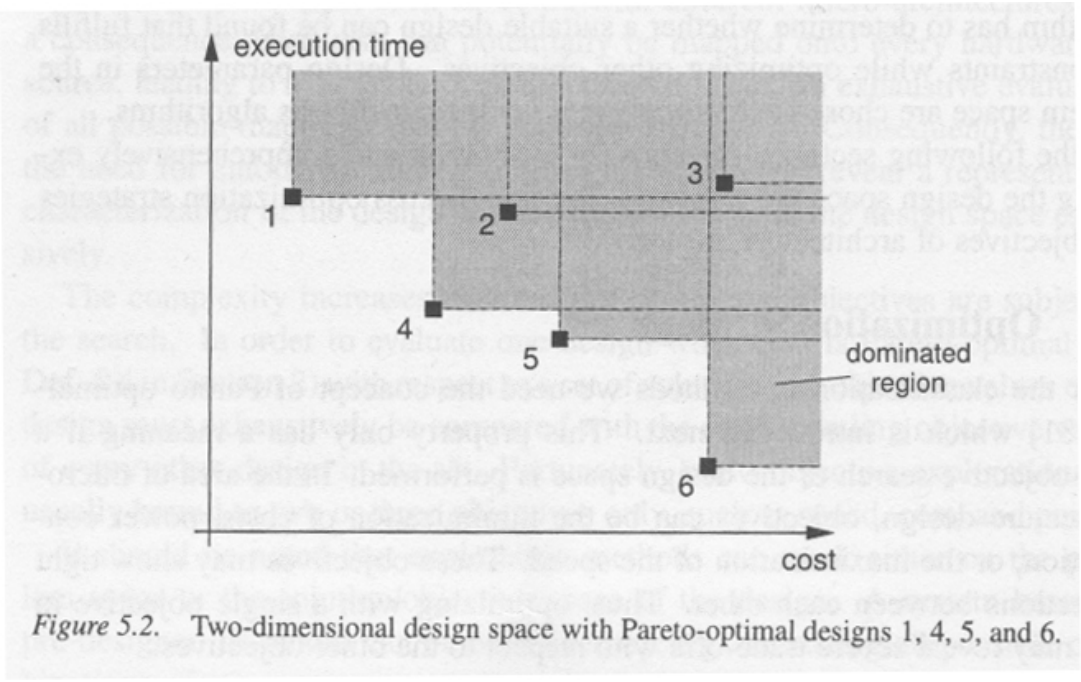


Figure 5.2. Two-dimensional design space with Pareto-optimal designs 1, 4, 5, and 6.

from Building ASIPS: The MESCAL Methodology by Gries, Keutzer, Meyr, Martin Springer Book

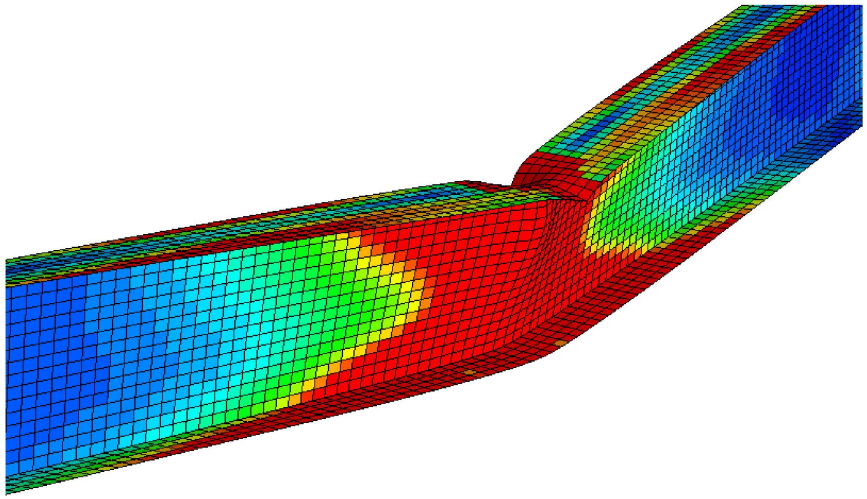
- decision making before search
  - aggregate different objectives into single objective before search is performed
  - convert certain objectives into constraints
- search before decision making
  - result is pareto optimal solutions
  - additional criteria are added to focus result
- decision making during search
  - iterative combination of above two
  - constraints can be determined automatically or interactively by presenting intermediate results

Architectural Performance

- energy
- strength
- cost
- weight
- size
- activity
- shade / sun
- acoustics
- lighting
- air flow
- etc

Finite Element Method

- hard to model strength of solid
- can approximate by using numerical integration
- divide solid into parts
- approximate function in each part
- tie together with linear equations
- solve linear equations



constraints are generally viewed as limiting factors in design, but there is strong evidence in architectural practice and research that constraints can trigger the development of innovative examples and are a powerful way to drive solution space (burrow and woodbury 1999).

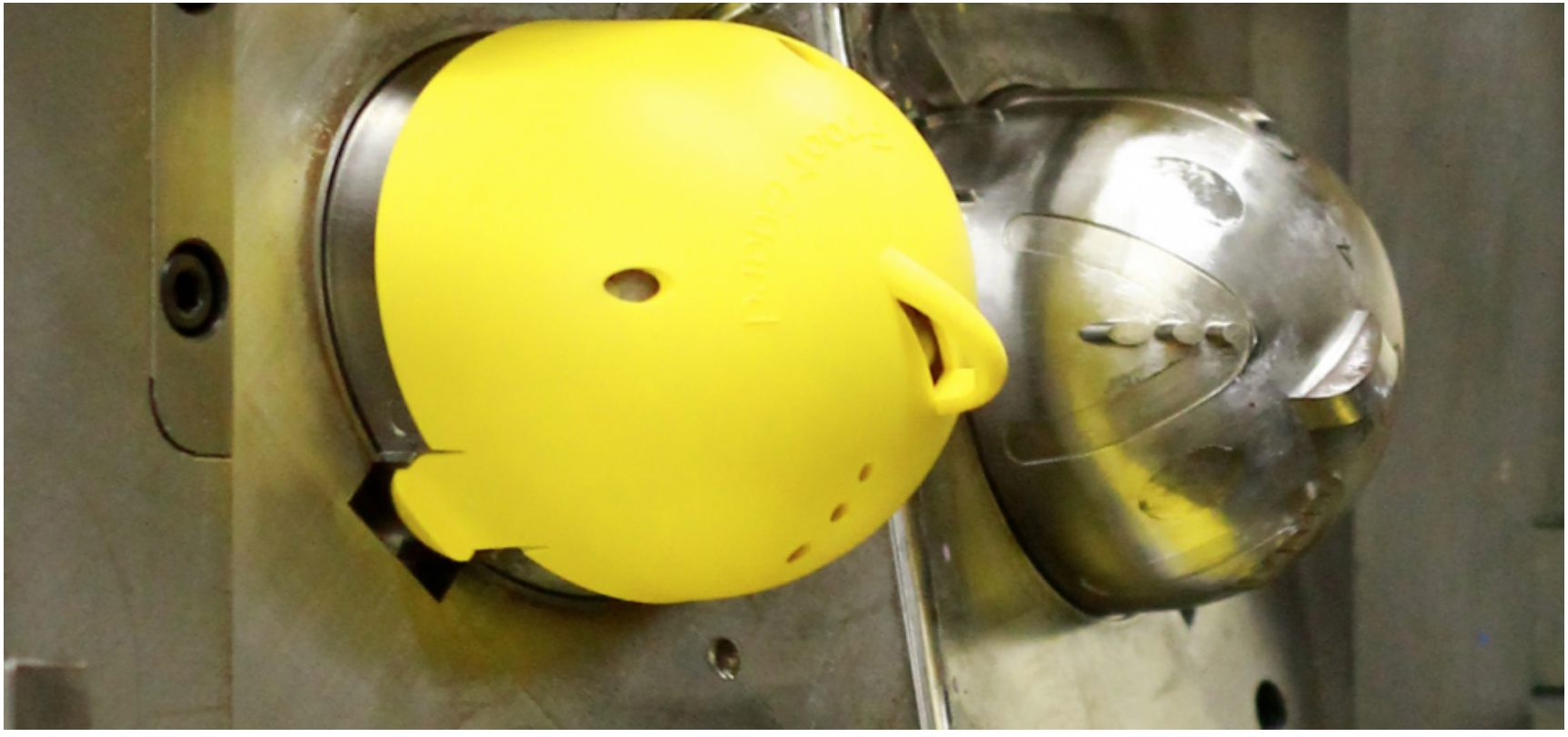
– axel kilian (design innovation through constraint modeling)

Kinds of Constraints

- functional – requirements for what system needs to accomplish functionally
- topologic – relationships between entities that form a topology
- geometric – geometric dimensions as well as relationships
- quantitative – measures such as volume, thickness, or length

Manufacturing Constraints

- developable pieces
- moldable
- 3 axis cuttable



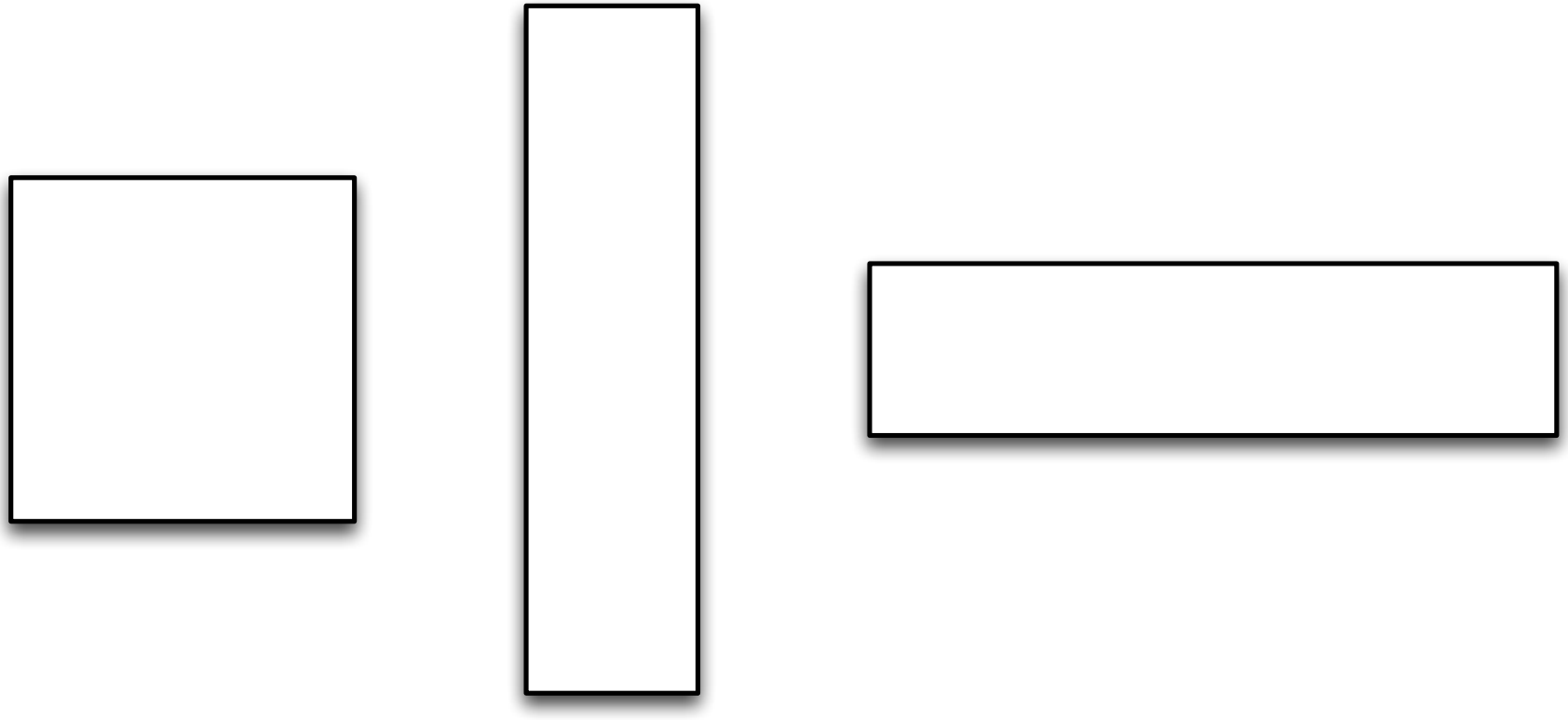
Design Rationale

- capture design decisions
- permit more thorough and principled design space exploration
- what are performance criteria?
- what are constraints?

Constraints


```
var a, w, h
always{ a == w * h }
```

- constant area
- pick two solve for third



Form Finding


```
var points[N]
always{ points[i] == avg_nbrs(i, points[k]) }
```

- can set some points
- solve for rest

Constraints

- relation between variables
- examples
  - function
    - 1-1 – $f = (c * 9/5) + 32$
    - n-1 – $x = 0$
  - arbitrary
    - n-n – $x < y$
    - n-n – $x = y + rand(-0.1, 0.1)$
- invertibility

- relationships between bodies
- kinds
  - hinges
  - universal
  - fixed
- could set up as feedforward but
- can set angles and find points or vice versa
- underconstrained – but can add regularizer

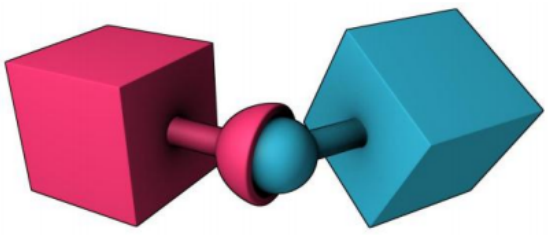


Figure 3 Point to point constraint

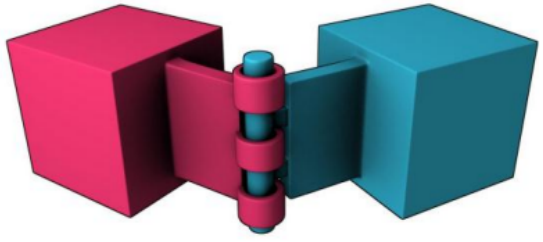


Figure 4 Hinge Constraint

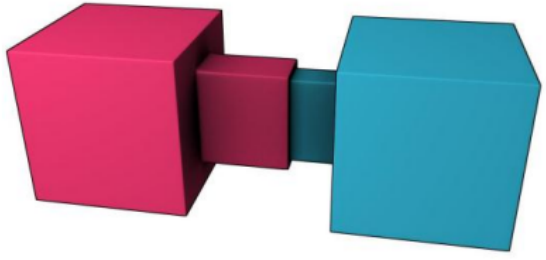


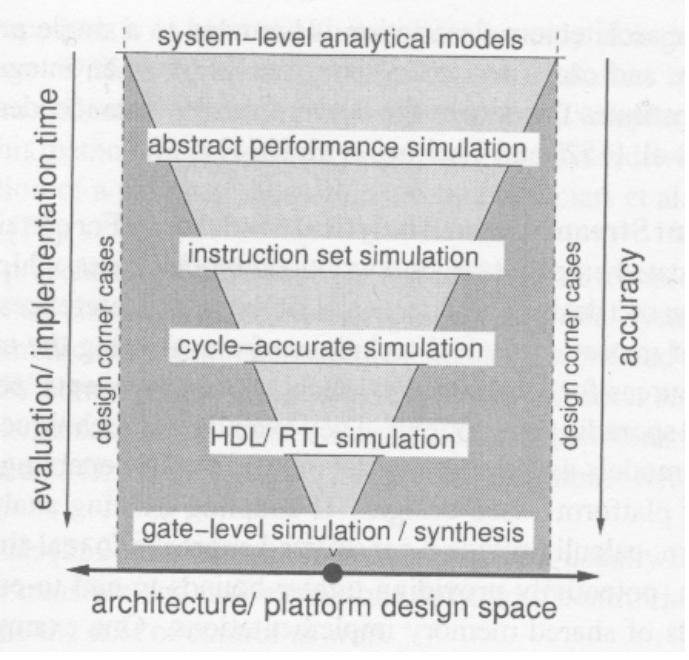
Figure 5 Slider Constraint

from Bullet physics simulation manual

Softness

- hard – can not violate
- soft – can violate with some penalty
- cost function – weight cost of violation

- simulation
  - system-level simulation
  - cycle-accurate simulation
- combined simulation/analytical
  - trace-based performance analysis
  - analytical models with calibrating simulation
- purely analytical
  - static profiling
  - event stream-based analytical models
  - high-level synthesis

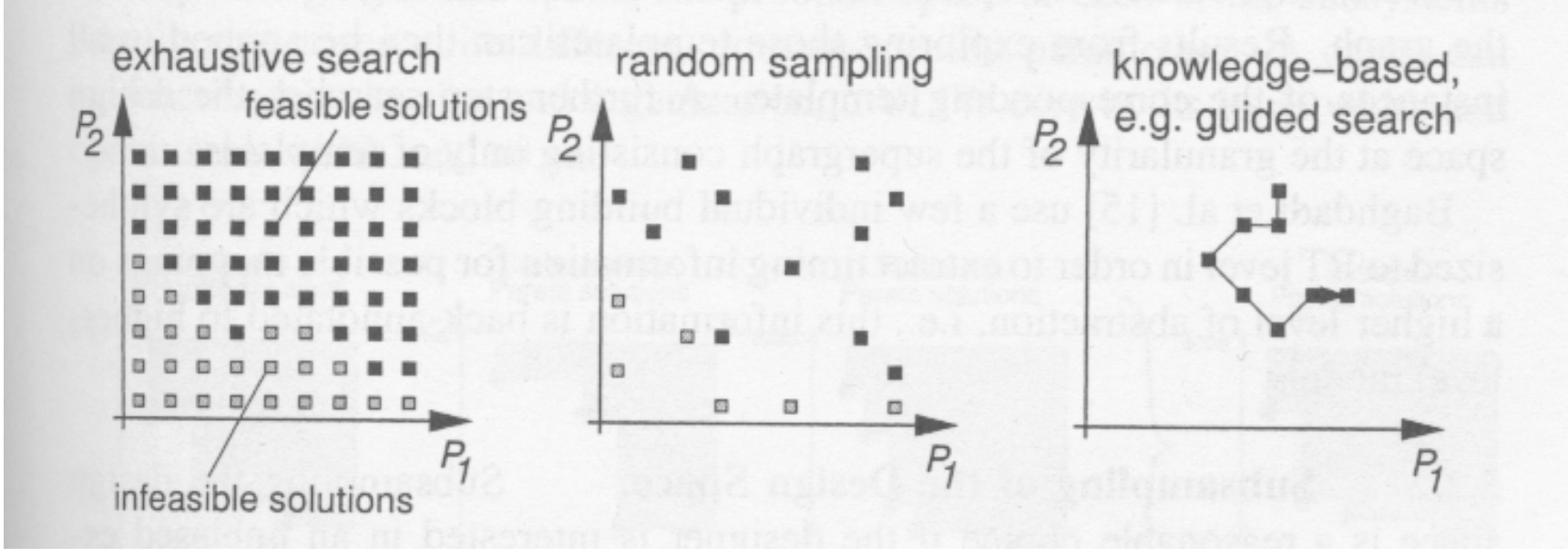


Cost Functions

- numeric mapping of desired properties
- often weighted sum of terms

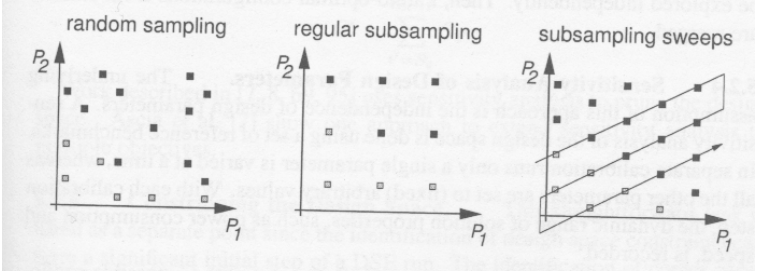
Exploration Techniques

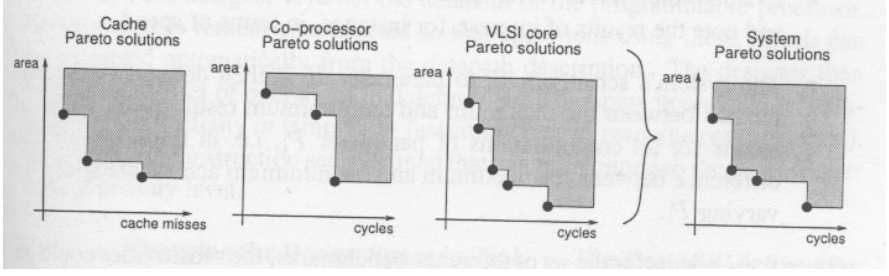
- exhaustive
- randomly sampling
  - monte carlo
  - simulated annealing
- guided search
  - hill climbing
  - evolutionary search
- ad hoc techniques



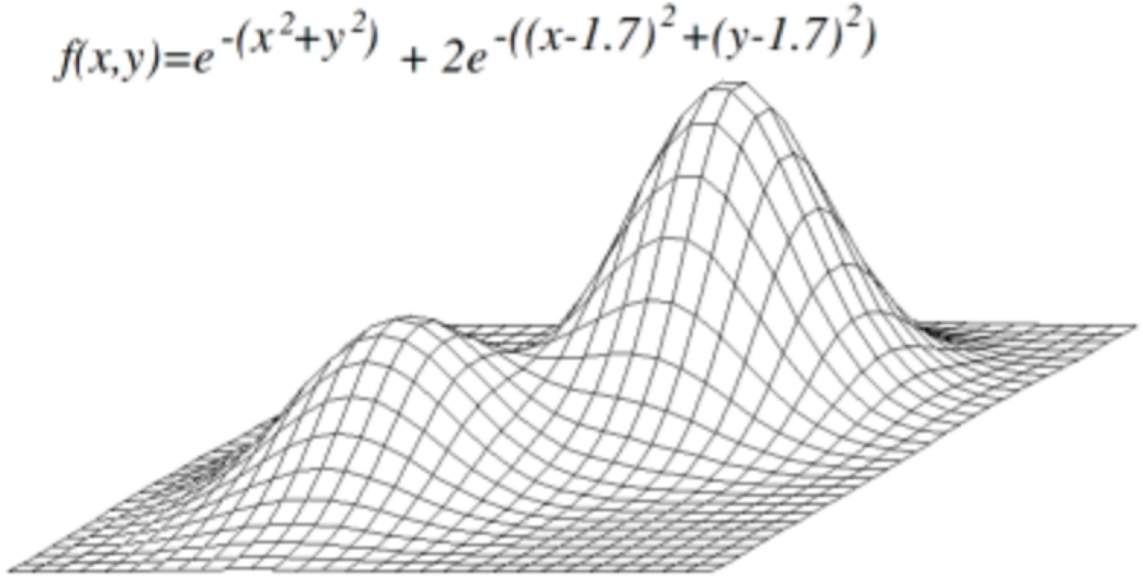
Pruning Techniques

- hierarchical exploration
- subsampling of the design space
- subdividing the design space into independent parts
- sensitivity analysis of design parameters
- constraining the design space





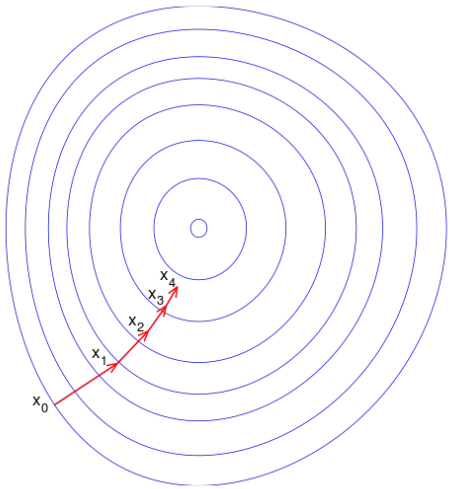
- incrementally find locally best solution
- subject to local minima



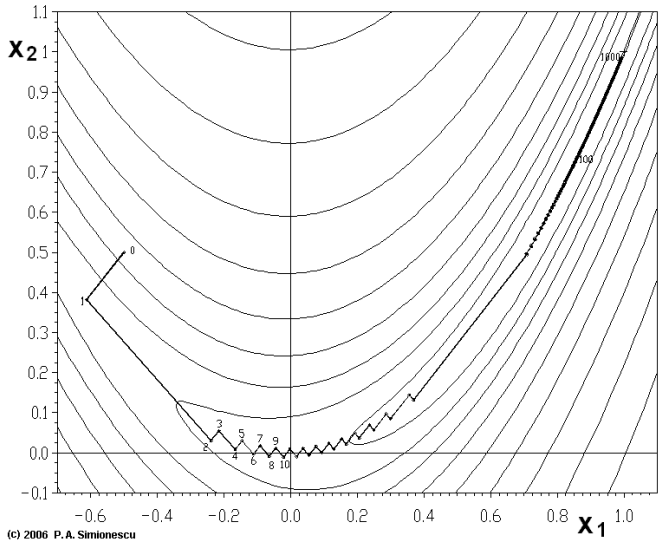
by Shashenka

Gradient Descent

- differentiate function wrt parameters
- adjust parameters a fraction of gradient
- need to differentiate function
- used to train neural networks with chain rule

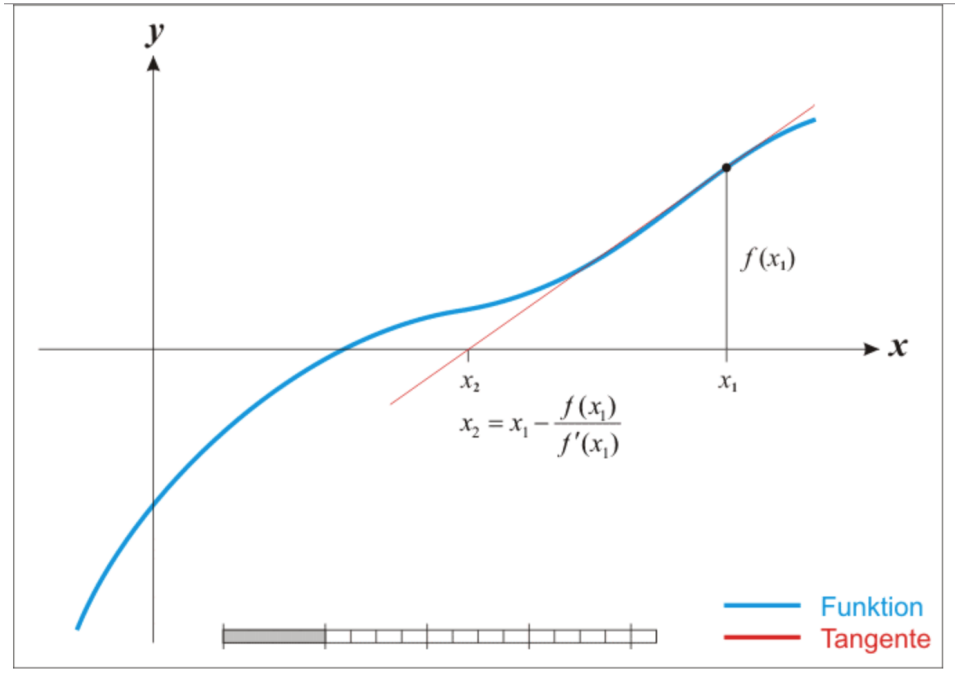


by Olegalexandrov



by Simionescu

- more expensive to compute step but can converge faster
- newton's method assumes locally quadratic

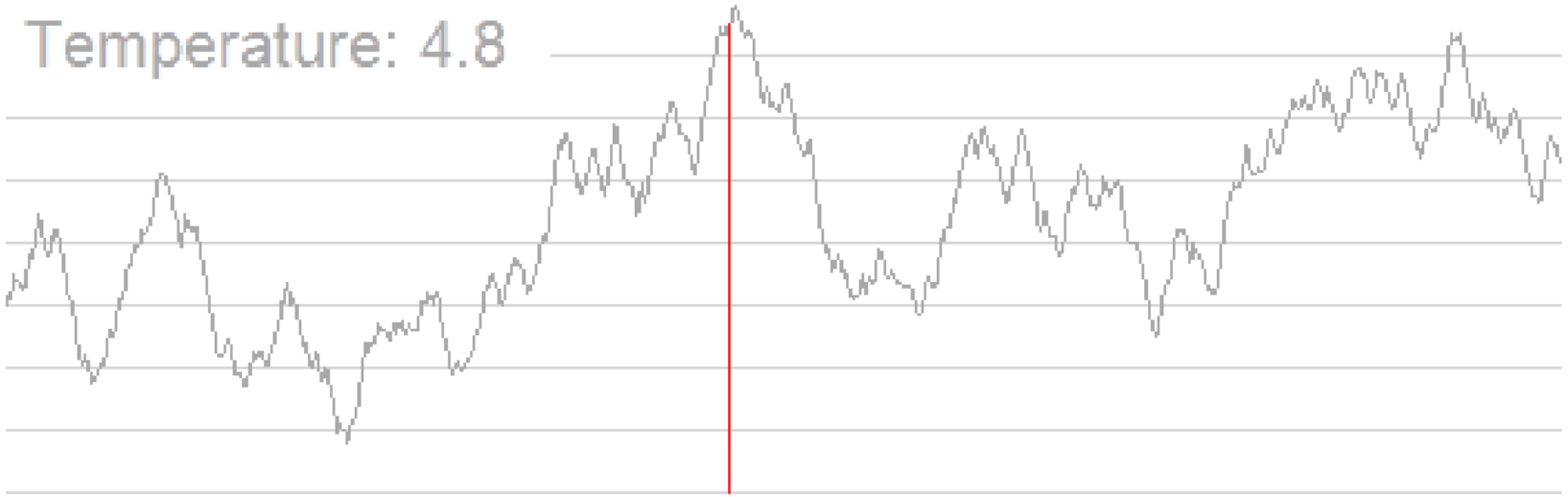


Estimate Gradient

- whole bunch of techniques for when you don't have derivative
- BFGS – uses many function evals to approx second derivative ...
- `scipy.optimize.minimize`

Simulated Annealing

- high to low temperature
- explore coarse to fine
- use moves to generate candidate neighbors
- cost function but no need for derivative
- can put in constraints

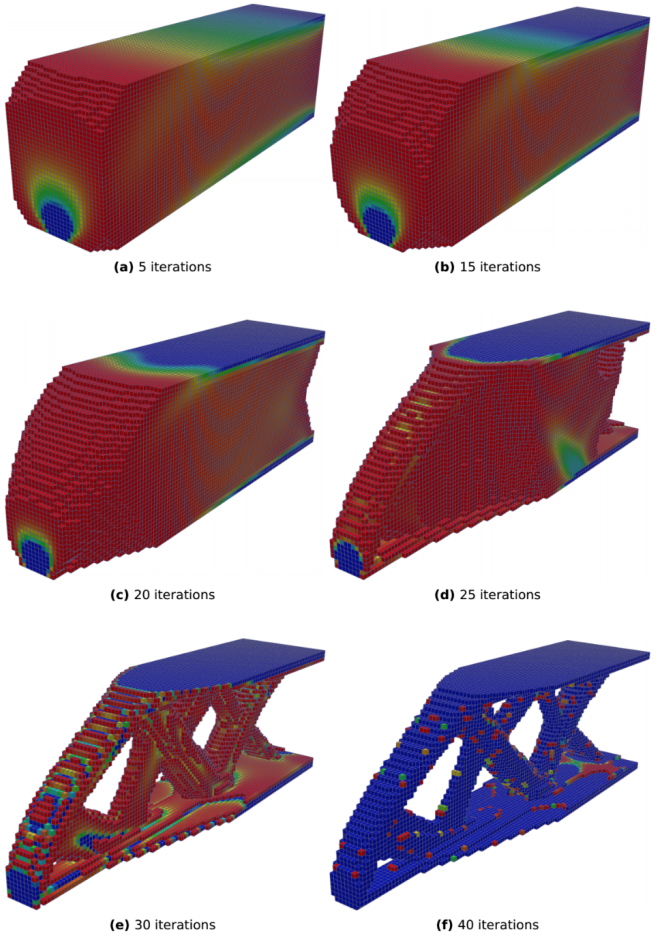


by kingpin13

- population of individuals
- choose best individuals to mate
- next generation created using mutation and cross over
- gives multiple solutions

by Hornby et al + NASA

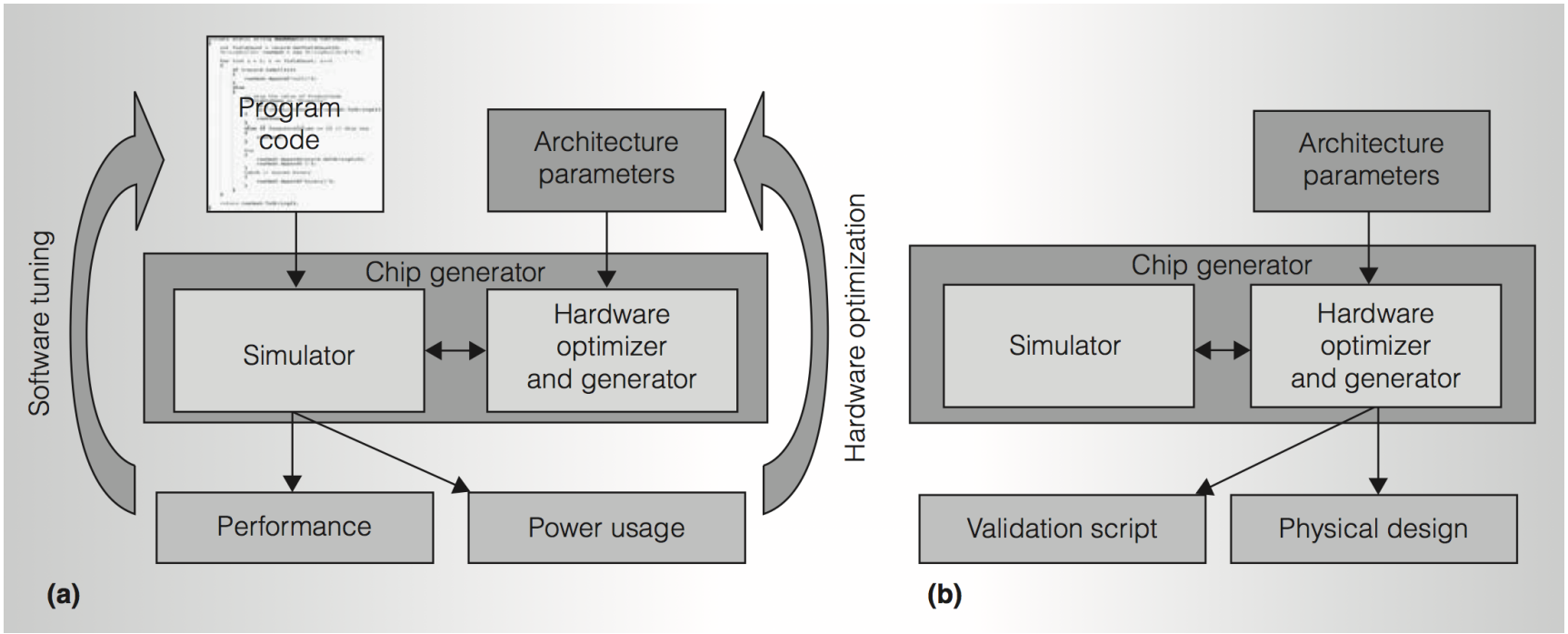
- topology finding
- based on finite element analysis
- optimization technique



by William Hunter

Exploration Workflow

- input energy budget
- simulation for performance
- ASIC workflow for energy usage
- fed back for refinement



from Rethinking Digital Design: Why Design Must Change by Shacham et al in IEEE Micro magazine

Bidirectional Constraints

- What design changes can bring about the desired performance?
- How much can a design parameter be changed while still maintaining performance in a given range?
- What are the various combinations of design parameters that yield the same performance in a given context?

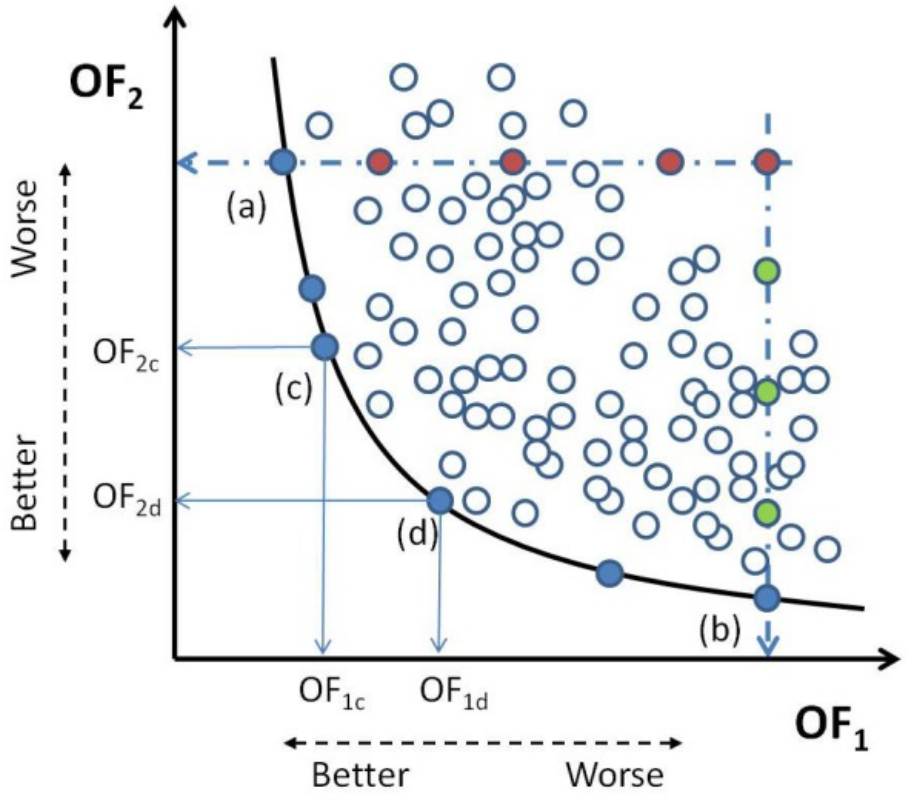
Bi-directional computational design support in the SEMPER environment – Mahdavi + Mathew + Kumar + Wong

- It will allow a designer to make desired changes in performance variables and observe the corresponding changes in design variables
- it will allow a designer to make changes in design variables and observe resulting changes in other design variables when one or more relevant performance variables are constrained.

Projection Based Approach


```
p = start with a initial design parameters
loop
  p = user changes design
  p = greedily improve design according to performance
until good enough
```

Bi-directional computational design support in the SEMPER environment – Mahdavi + Mathew + Kumar + Wong


OF2
Worse
Better
OF2c
OF2d
(a)
(c)
(d)
(b)
OF1c
OF1d
OF1
Better
Worse

Ordered Exploration

```
always{ a < 10 }
minimize{ energy(a, b) }
minimize{ speed(a, b) }
```

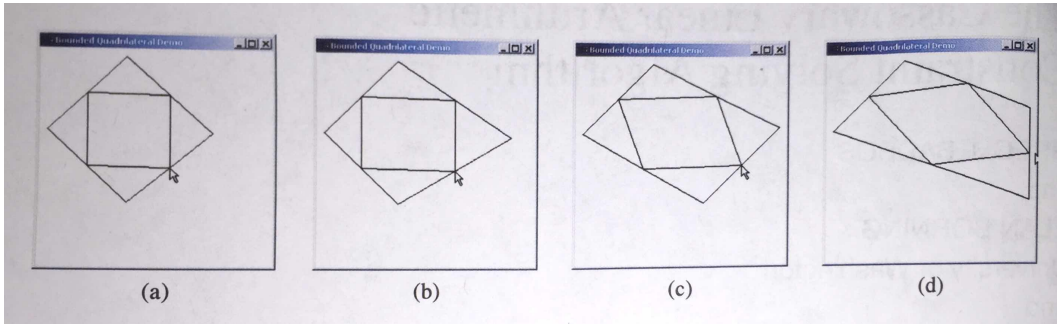
- evaluation methods have varying speed
- order satisfaction based on speed

- automatically propagate changes between link variables
- use always construct which returns boolean
- can keep everything connected
- can set up more interesting algebraic relationships

```
@f = 0, @c = 0
always{ f = 9/5 * c - 32 }
```

- use always construct which returns boolean
- keeps all assertions true

```
@mid[N] = (10,10), @pt[N] = (10,10)
always{ mid[i] = 0.5 * (pt[i] + pt[i+1]) }
always{ pt[i].x >= 10 and pt[i].y >= 10 }
```

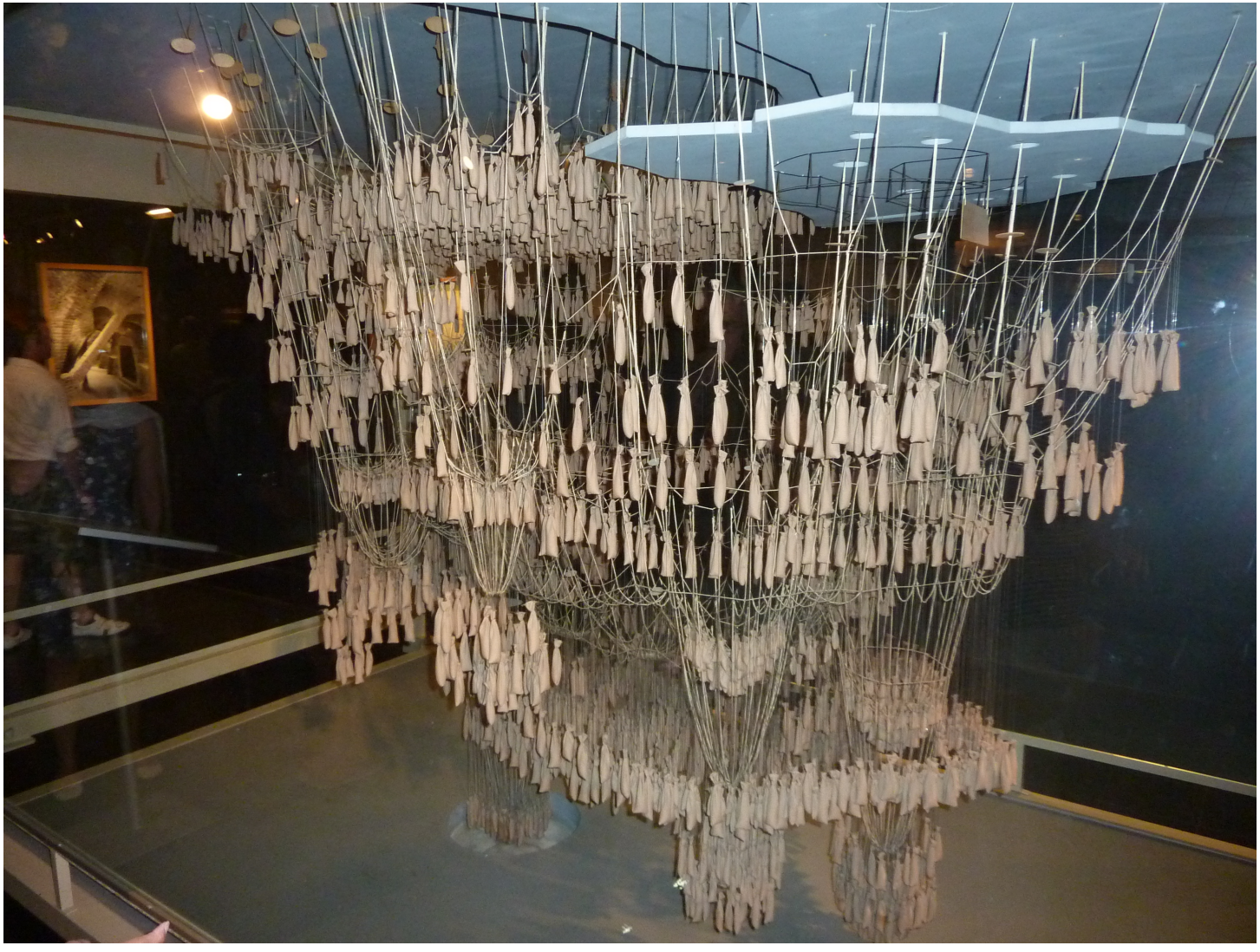


```
@a = 0, @b = 0
minimize{ energy(a, b) }
```

- How do you work backwards?
- If system is invertible great
- Even if differentiable then that works
- Otherwise need to do search

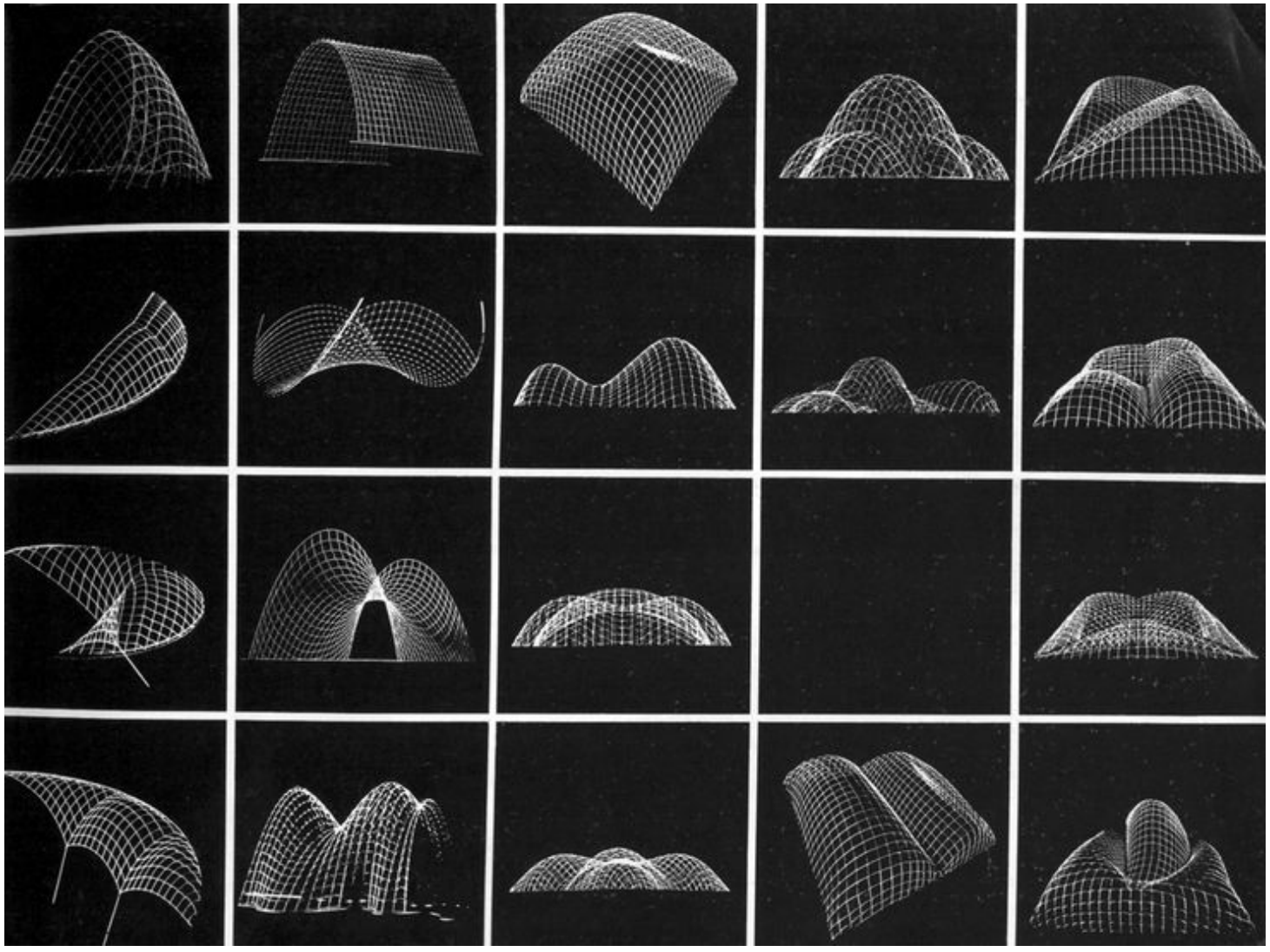
- Constraints
- Performance
- Find best

Gaudi – Catenary Structures

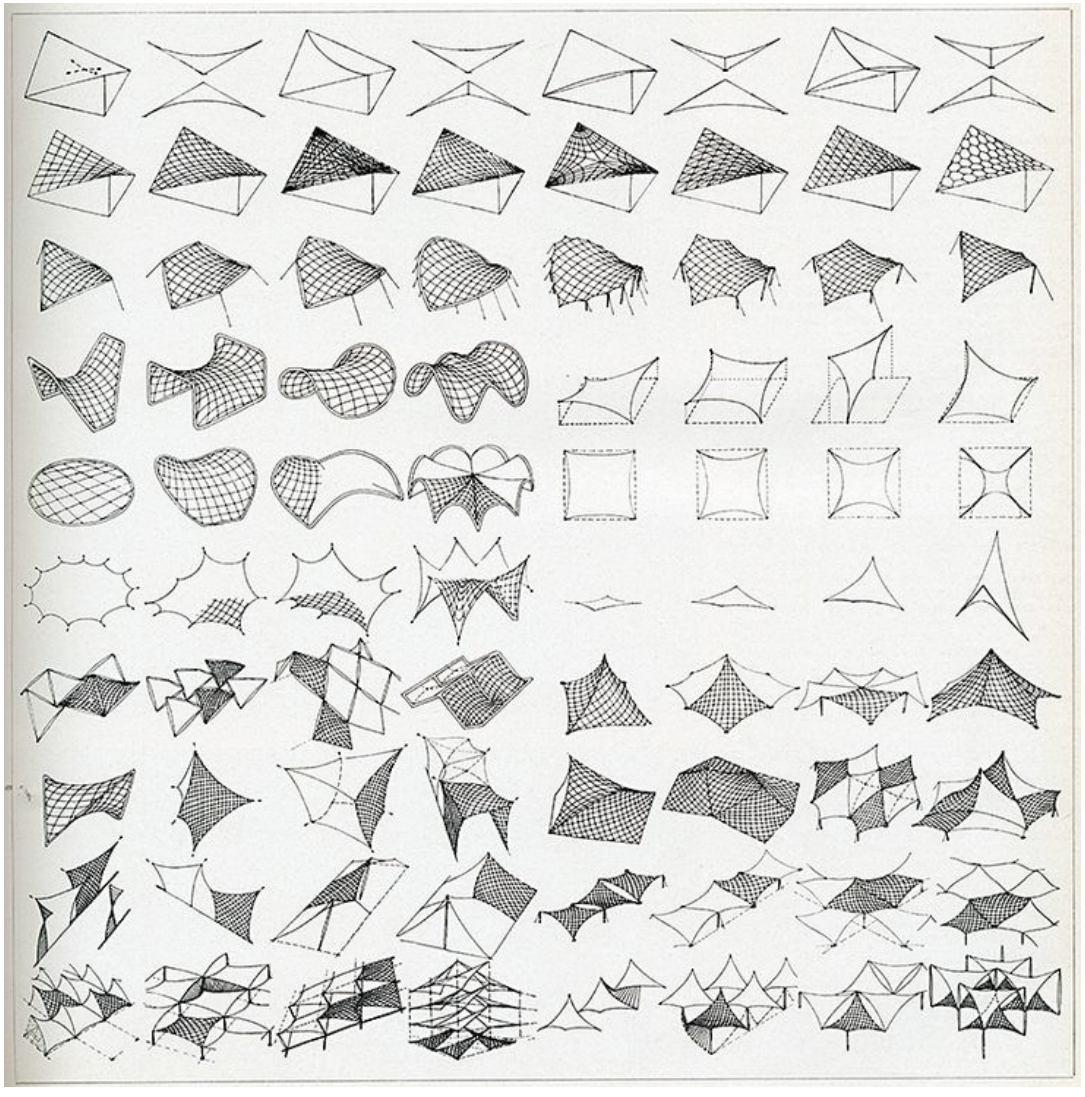


Gaudi – Colonia Church

Frei Otto – Form Finding

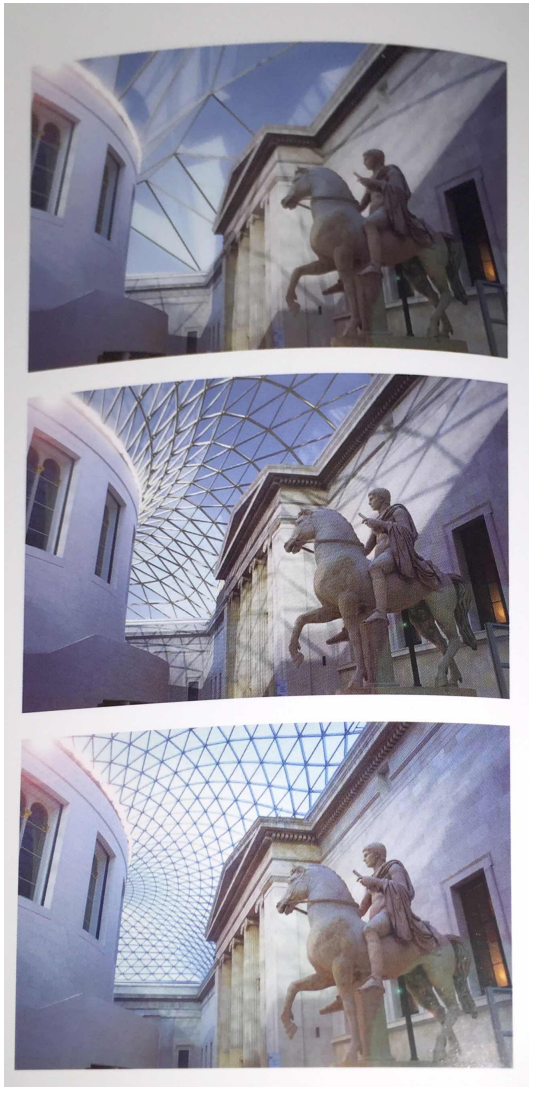


Frei Otto – Studies



Frei Otto – Train Station

Court – Studies



British Museum Great Court

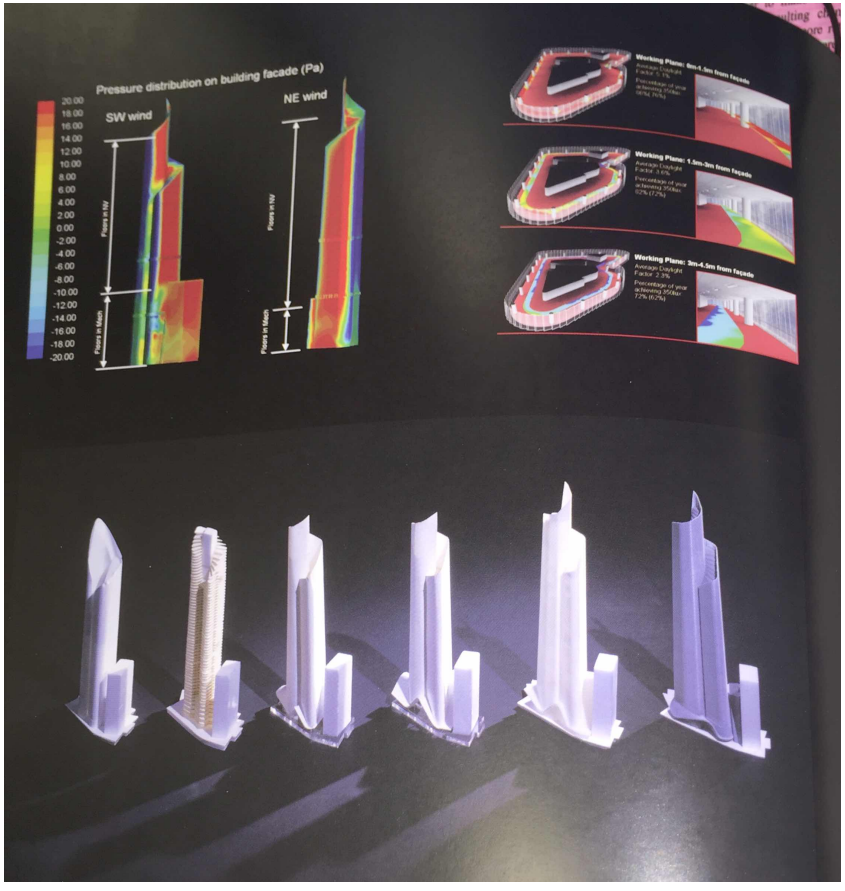
Foster + Partners, London UK

Pinnacle – Kohn Pedersen Fox

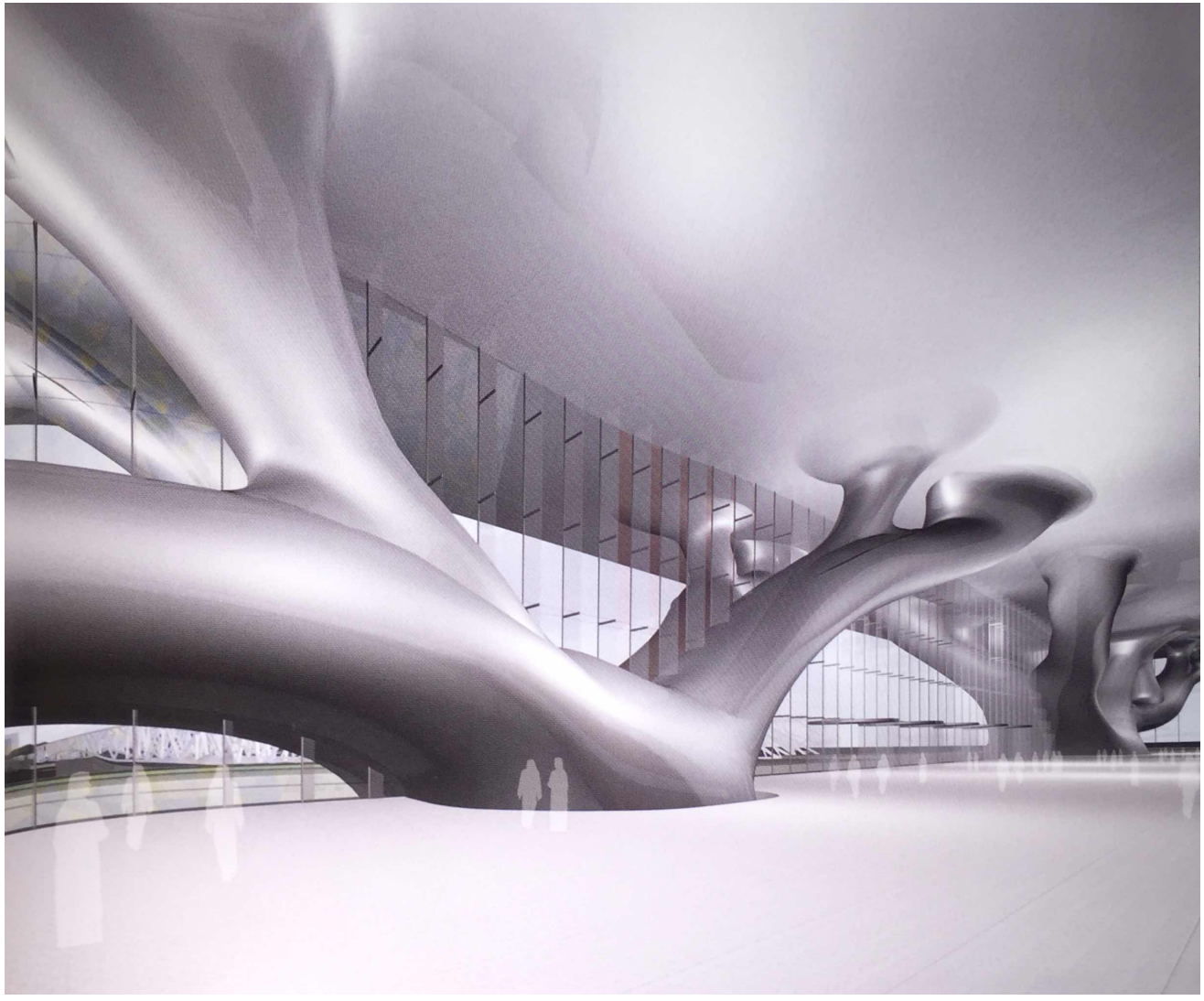
London, UK

Pinnacle – Models

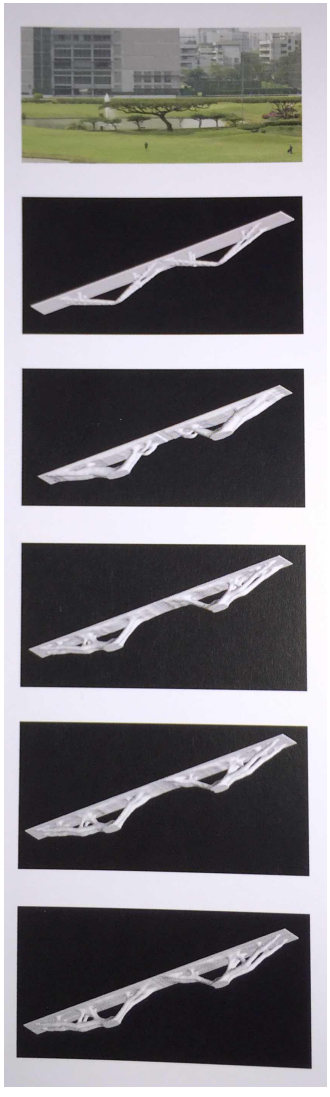
Pinnacle – Studies

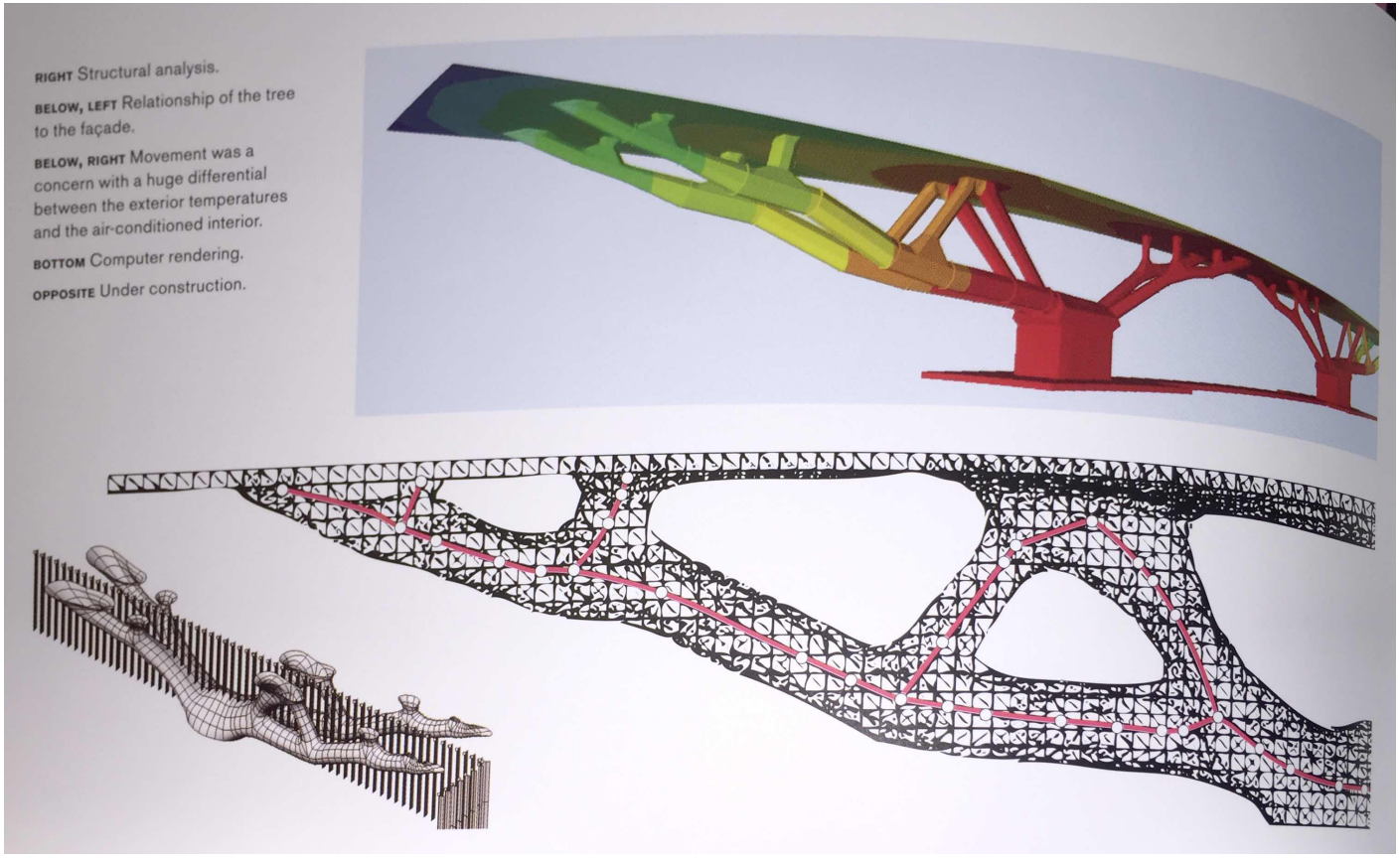


Qatar Education City Convention Centre



– Arata Isozaki, London, UK





RIGHT Structural analysis.

BELOW, LEFT Relationship of the tree to the façade.

BELOW, RIGHT Movement was a concern with a huge differential between the exterior temperatures and the air-conditioned interior.

BOTTOM Computer rendering.

OPPOSITE Under construction.

Next Week: Generative Design + Future Fab

Robotic Brick Fabrication | EVSC

- *Design Innovation through Constraint Modeling* by Axel Kilian
- *Babelsberg: Specifying and Solving Constraints on Object Behavior* by Felbentreff + Borning + Hirschfeld
- *Bi-directional computational design support in the SEMPER environment* by Mahdavi + Mathew + Kumar + Wong
- *The New Mathematics of Architecture* by Burry + Burry
- *Spec2Fab: A Reducer-Tuner Model for Translating Specifications into 3D Prints* by Chen + Levin + Didyk + Sitthi-Amorn + Matusik