Optimization Methods: an Applications–Oriented Primer

Antonio Frangioni

Dipartimento di Informatica, Università di Pisa

with L. Galli

Analytics for the Sharing Economy:
Mathematics, Engineering and Business Perspectives

Workshop within the
European Control Conference '19

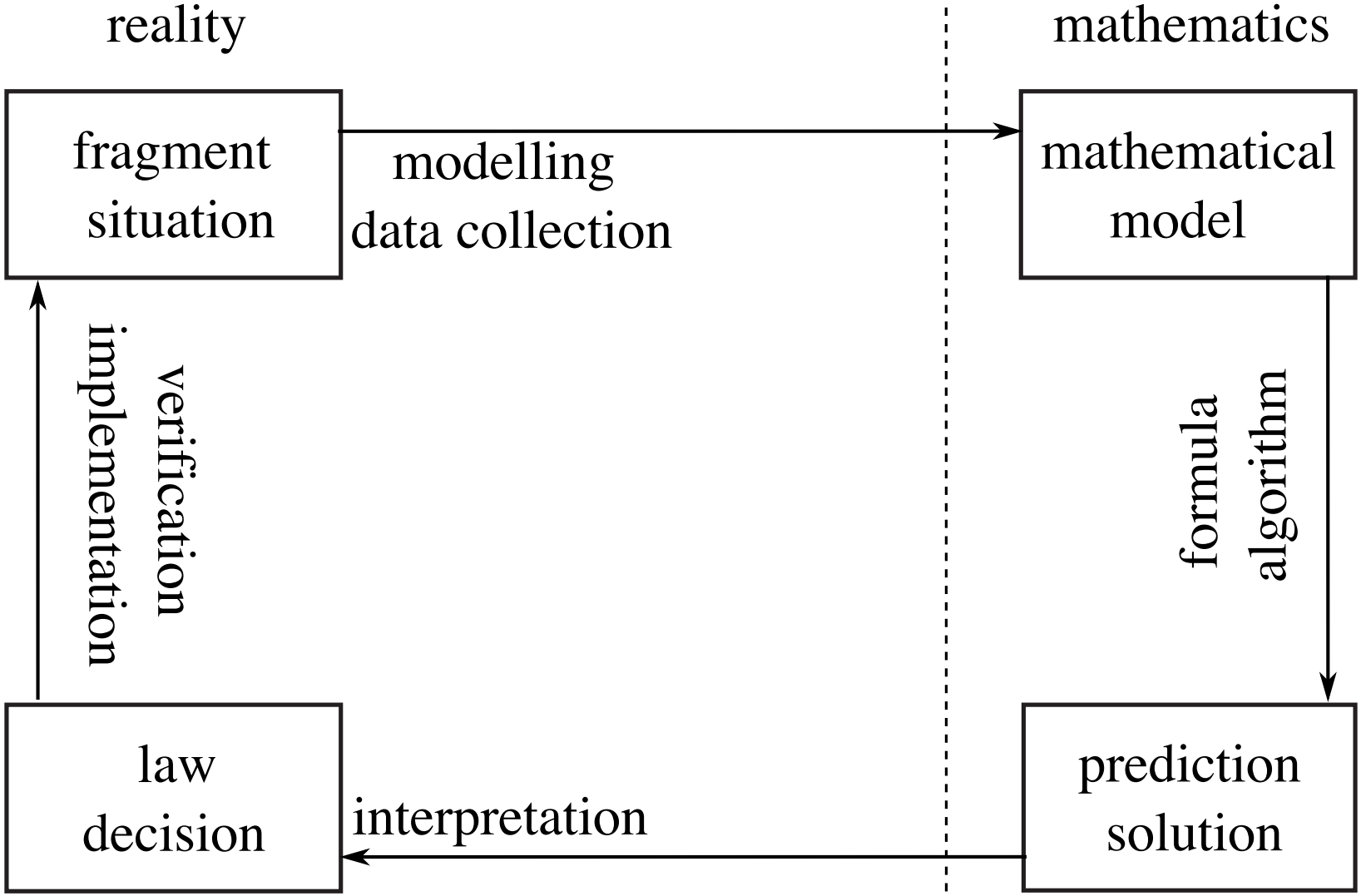
Napoli – June 25, 2019

Outline

Why Optimization in the Sharing Economy?

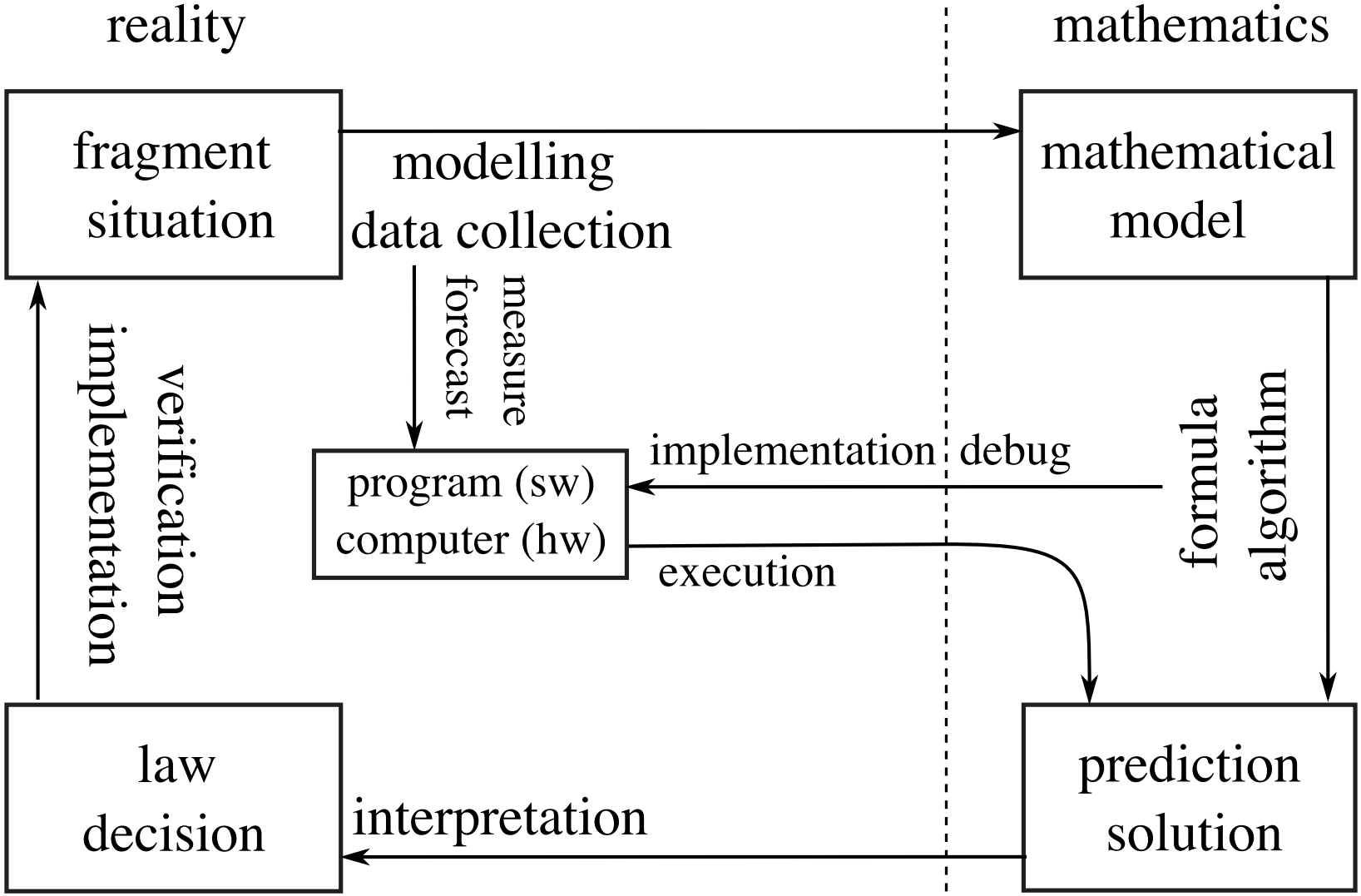
- Sharing Economy just makes economical sense:
  too many costly items/services are under-utilized
- Why has it traditionally always been like this?
  - Cultural reasons (strangers in my house/car . . .)
  - It is just way simpler: my stuff is always there when I need it
- Realising the economic potential of a shared good is nontrivial
- Some cases simple enough: fixed goods with long utilisation times,
  then efficient discovery is all you need (and the web gives it)
- Very complex in other cases: mobile goods with short utilisation times
- Technology makes it possible, but logistics a significant issue
- Shared utilisation convenient only if system optimally managed
- How can this be done? Primarily via a mathematical model

Mathematical models



- The fundamental cycle

Mathematical models



- The fundamental cycle and its implementation

Outline

Optimization Problems

- Descriptive model: how the world (supposedly) is $\equiv$ complex
  - no closed-form description $\mapsto$ simulation, and/or
  - complex description (ODE/PDE constraints, ...), and/or
  - multiple decision makers (equilibria, ...), and/or
  - uncertainty in data (forecasts, ...)
- Prescriptive model a.k.a. optimization problem: how the world should be

$$(P) \quad f_* = \min \left\{ f(x) \ : \ x \in X \right\}$$

  - arbitrary set X = feasible region of possible choices x
  - $f : X \to \Re$ (simple case) objective function mapping preferences

  of course even harder, in fact impossible
- f and X must be "nice" to have any chance of success
- At least "$f(x)$", "$x \in X$" computable in black-box form
- Black-box optimization possible, but slow (curse of dimensionality)
- Large-scale optimization requires exploiting structure

Explicit Algebraic Form of Objective and Constraints

- For a start: explicit form of objective and constraints

  $$\min \left\{ f(x) \ : \ g_i(x) \leq 0 \ \ i \in \mathcal{I} \right\}$$

  f, g_i given by (complex) algebraic expressions

- Still rather far from being manageable:
  - underlying space can be function space: ODE/PDE constraints
  - $\mathcal{I}$ can be infinite (uncountable)

- Most often infinite objects need be finitized:
  - no closed formulæ for ODE/PDE $\Longrightarrow$ discretization of x
  - approximation of X with dynamic finite $\bar{\mathcal{I}} \subset \mathcal{I}$

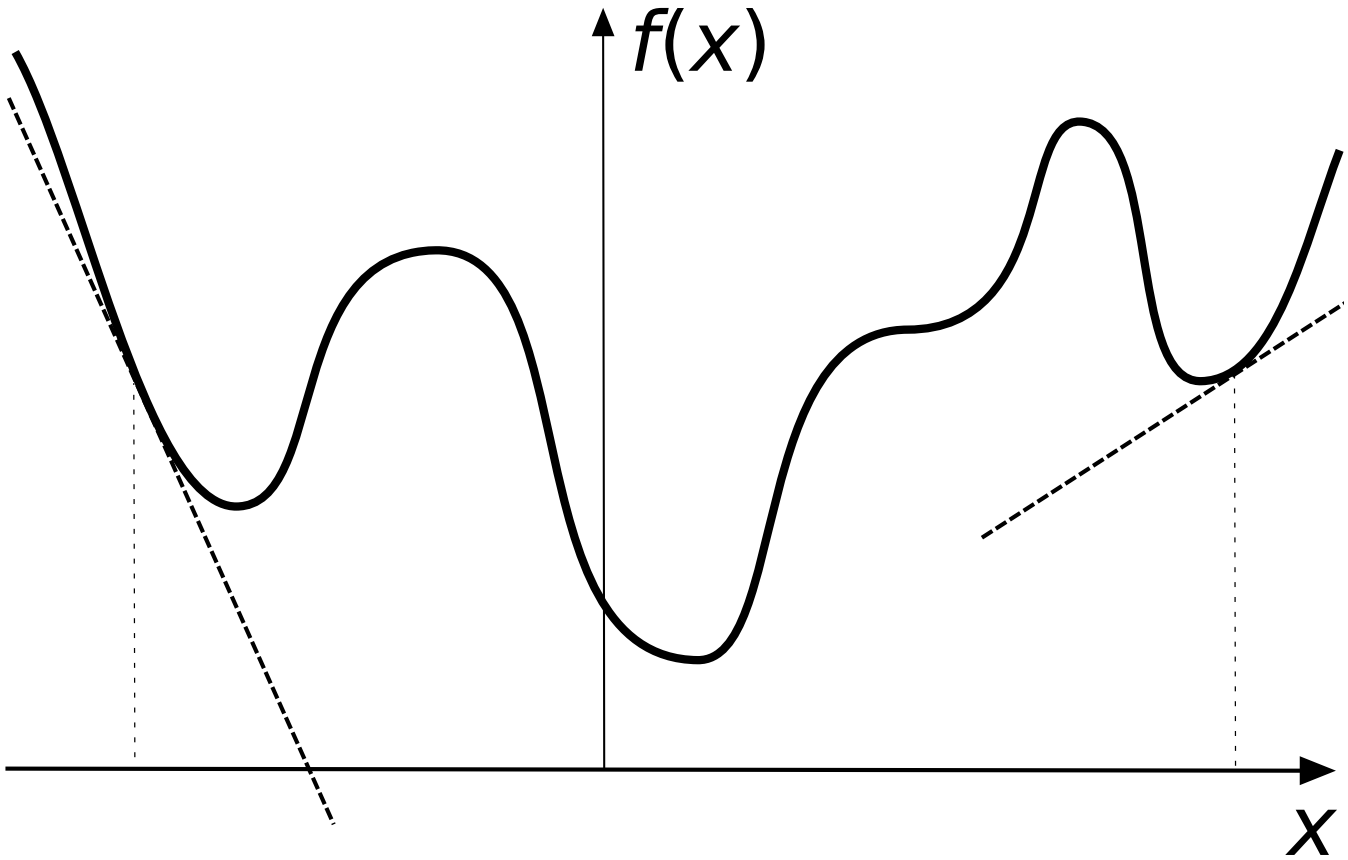
- Finite problem: $G(x) = [\ g_i(x)\ ]_{i \in \mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$, $f : \mathbb{R}^n \to \mathbb{R}$

- Still very hard: indecidable in general on unbounded region

  even for "simple" G, when decidable typically $\mathcal{NP}$-hard

Why Are Optimization Problems Hard?

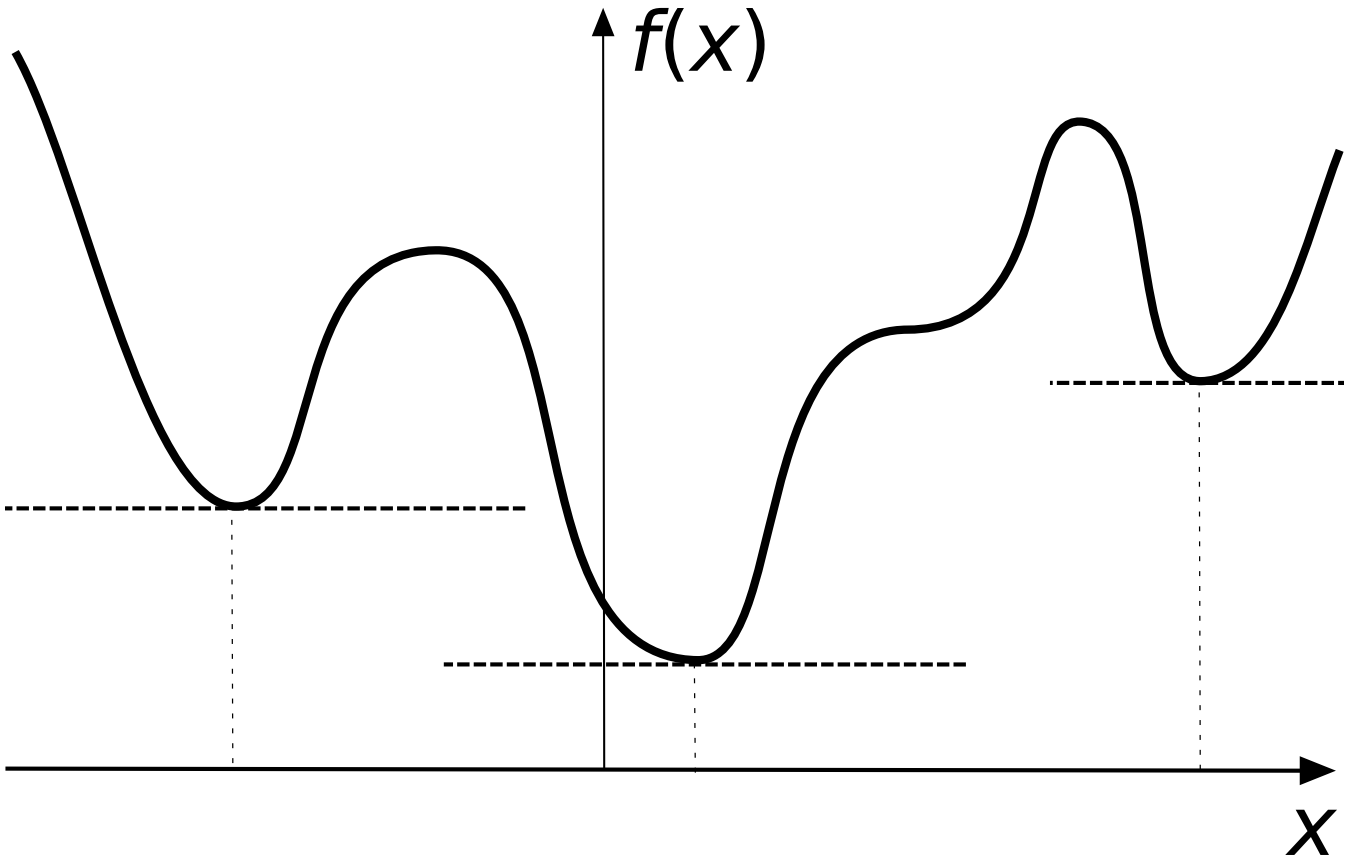
- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$

Why Are Optimization Problems Hard?

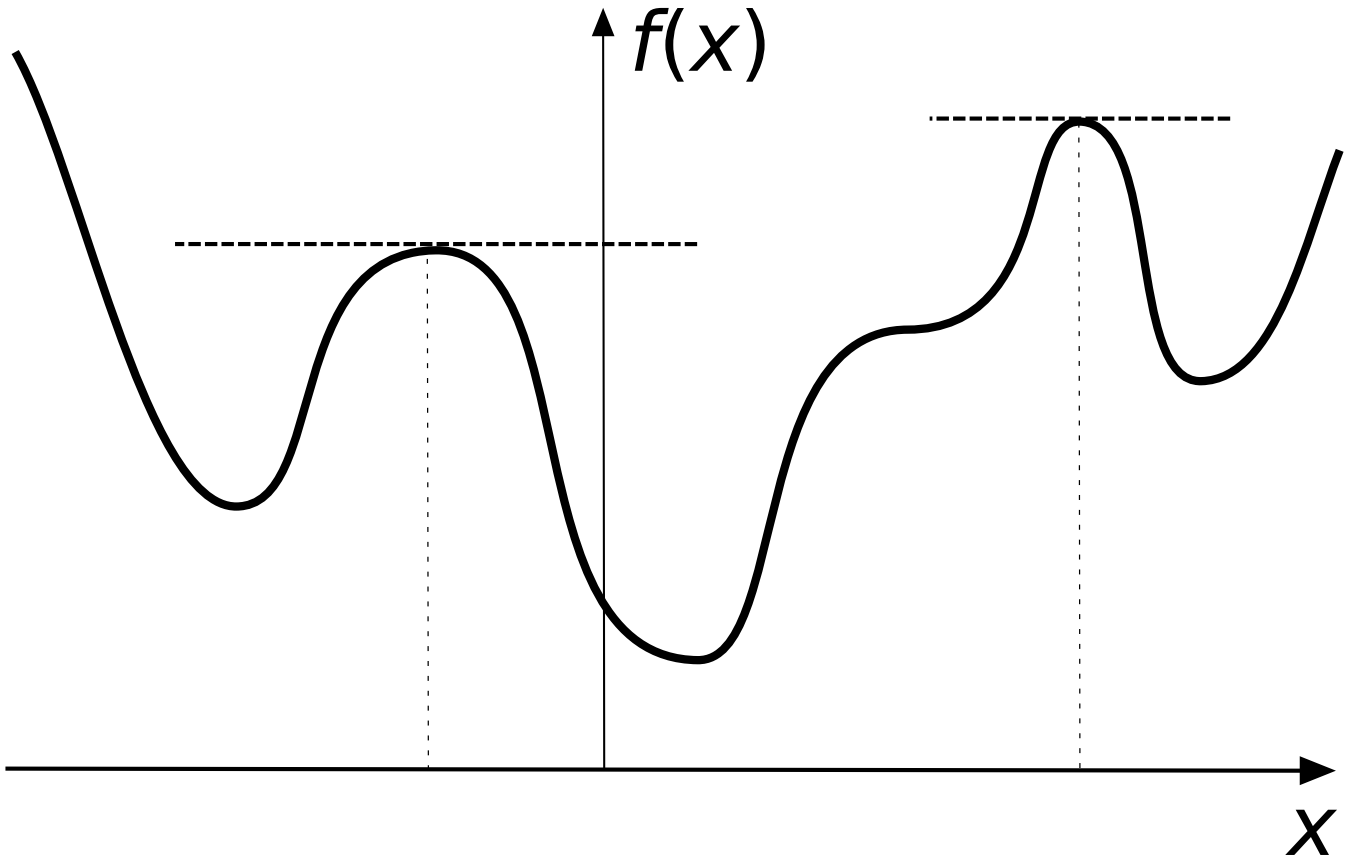
- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$
- $f'(x) = 0$ in all (local) minima,

Why Are Optimization Problems Hard?

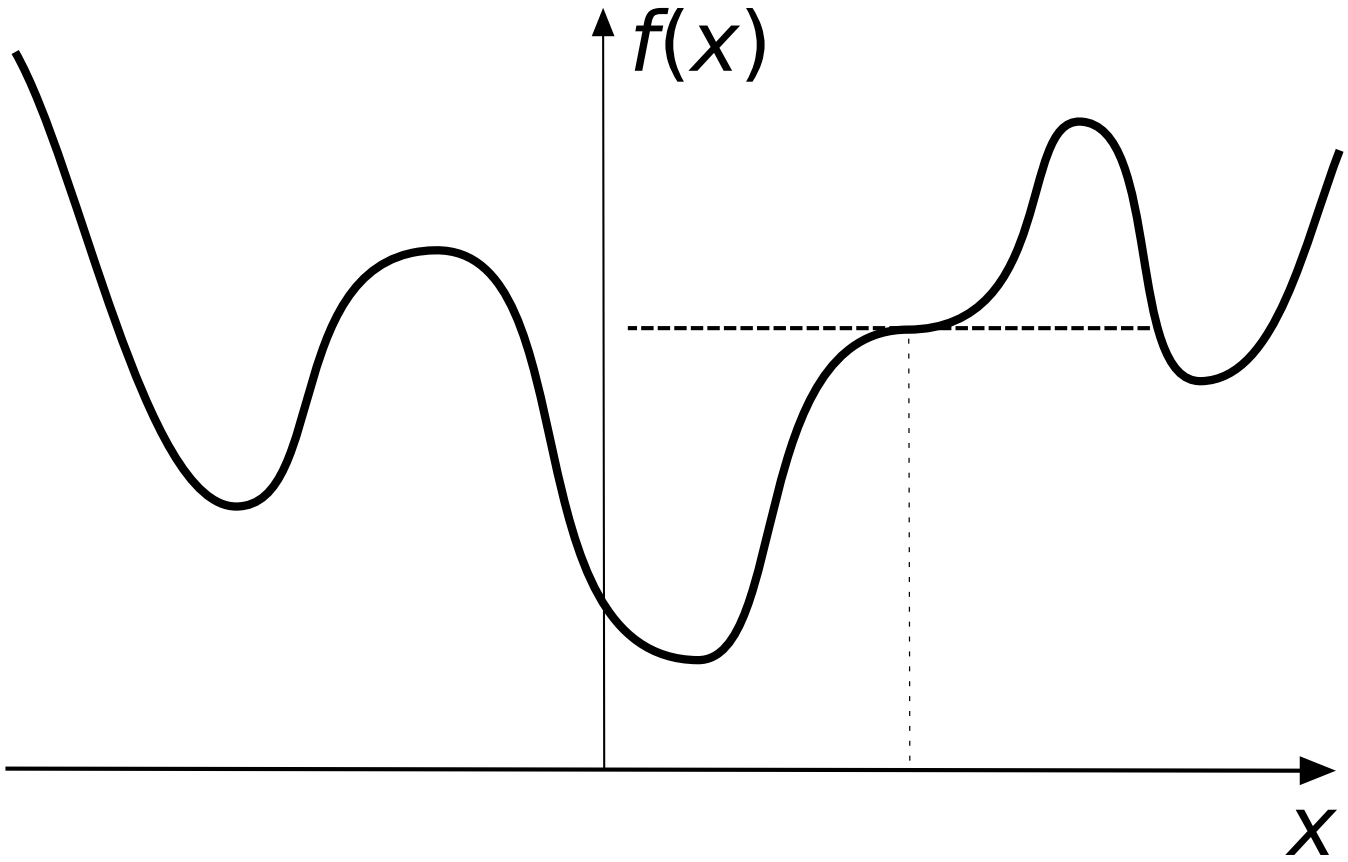
- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$
- $f'(x) = 0$ in all (local) minima, but also in local maxima

Why Are Optimization Problems Hard?

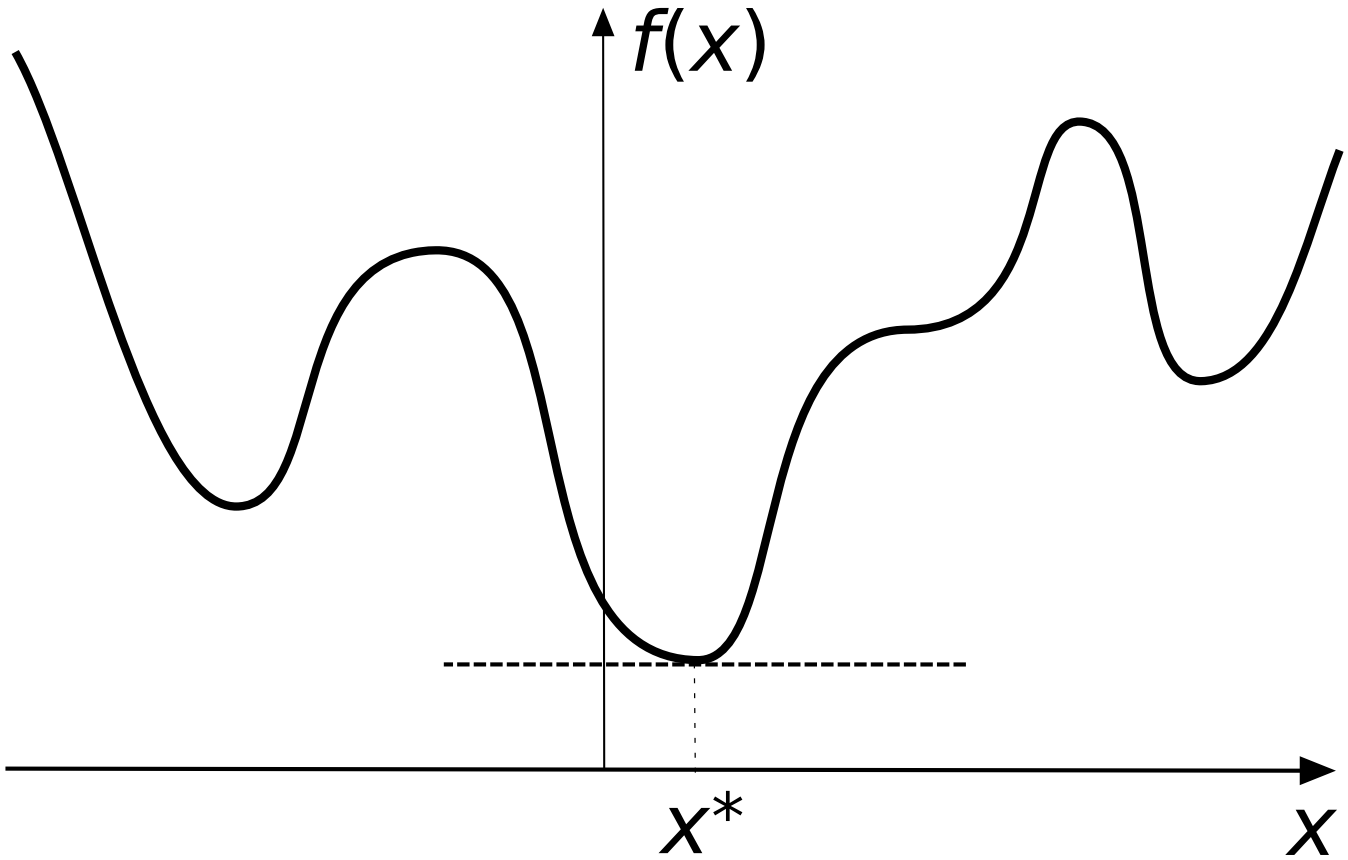
- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$
- $f'(x) = 0$ in all (local) minima, but also in local maxima & saddle points
- But all local minima "look equal", how do I

Why Are Optimization Problems Hard?

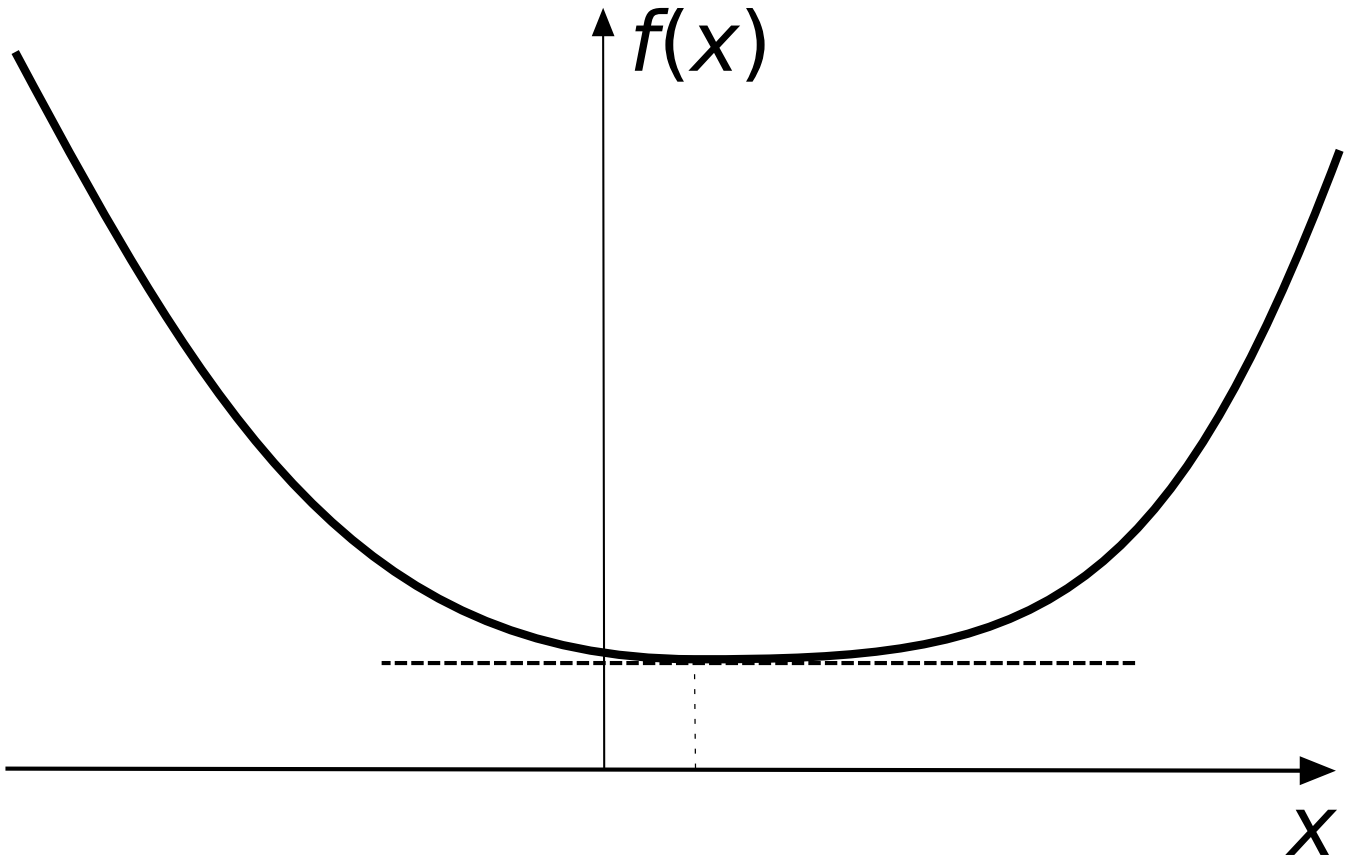
- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$
- $f'(x) = 0$ in all (local) minima, but also in local maxima & saddle points
- But all local minima "look equal", how do I find the global one?
- Only easy case:

Why Are Optimization Problems Hard?

- In fact, nobody knows ($\mathcal{P} \neq \mathcal{NP}$??), but intuitively



- Easy to prove that x not (locally) optimal: $f'(x) \neq 0$
- $f'(x) = 0$ in all (local) minima, but also in local maxima & saddle points
- But all local minima "look equal", how do I find the global one?
- Only easy case: no local maxima $\Longrightarrow$ f convex function

Convex Problems Are The Best!

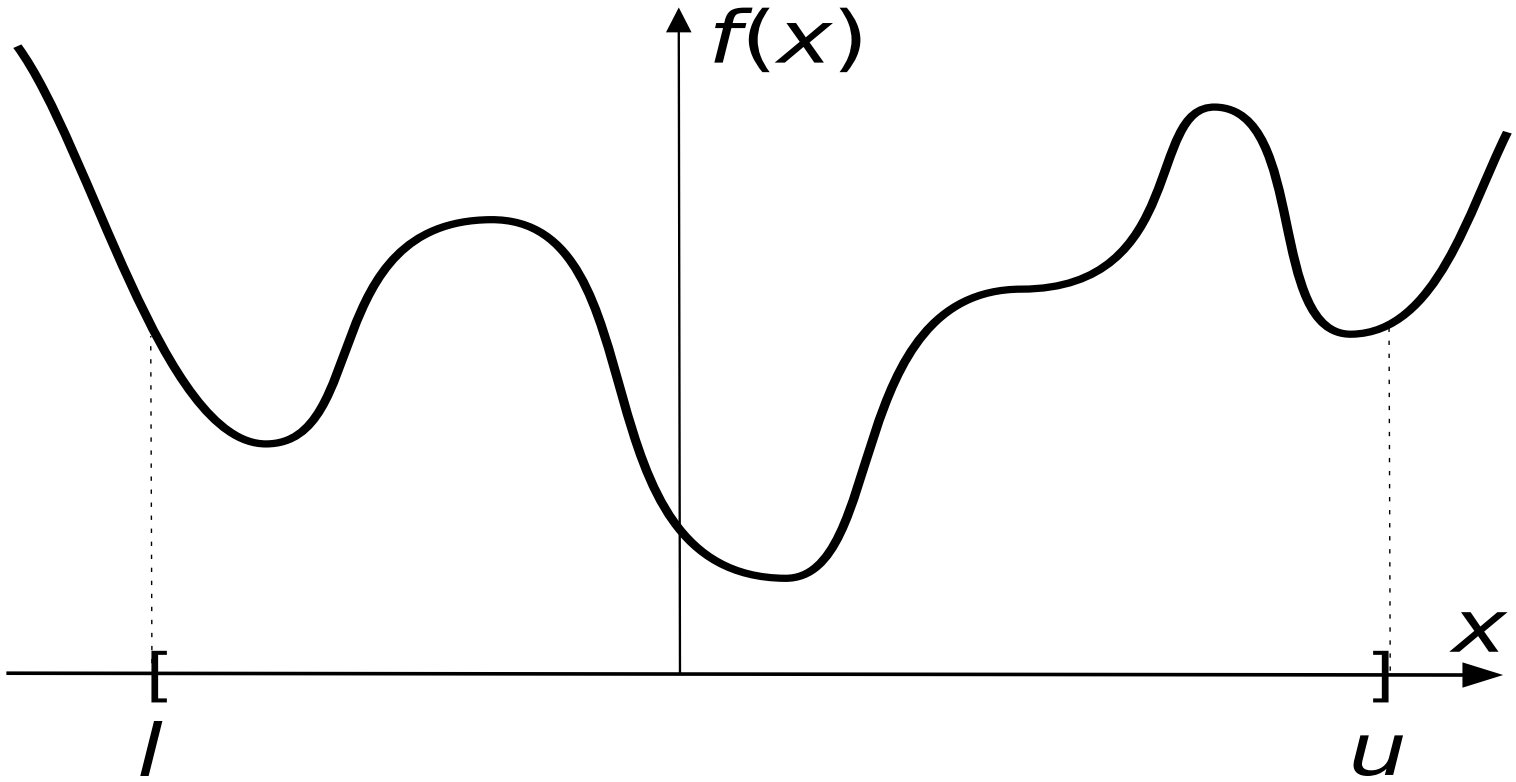
- $f, g_i : \mathbb{R}^n \to \mathbb{R}$ convex functions, not necessarily algebraic

  $$\min \left\{ f(x) \ : \ g_i(x) \leq 0 \ \ i \in \mathcal{I} \right\}$$

  but efficiently computable (and n, $|\mathcal{I}|$ "not too large")
- Can in general be solved "efficiently" (although it varies a lot)
- What can I solve with that? A lot, but by far not all
- Sometimes can assume convexity (can choose my model, e.g., ML)
- Sometimes local optima are fine as well (and not even these, again ML)
- Convex functions a rich family, some problems naturally convex
- But most problems are not convex at all!
- What do I do?

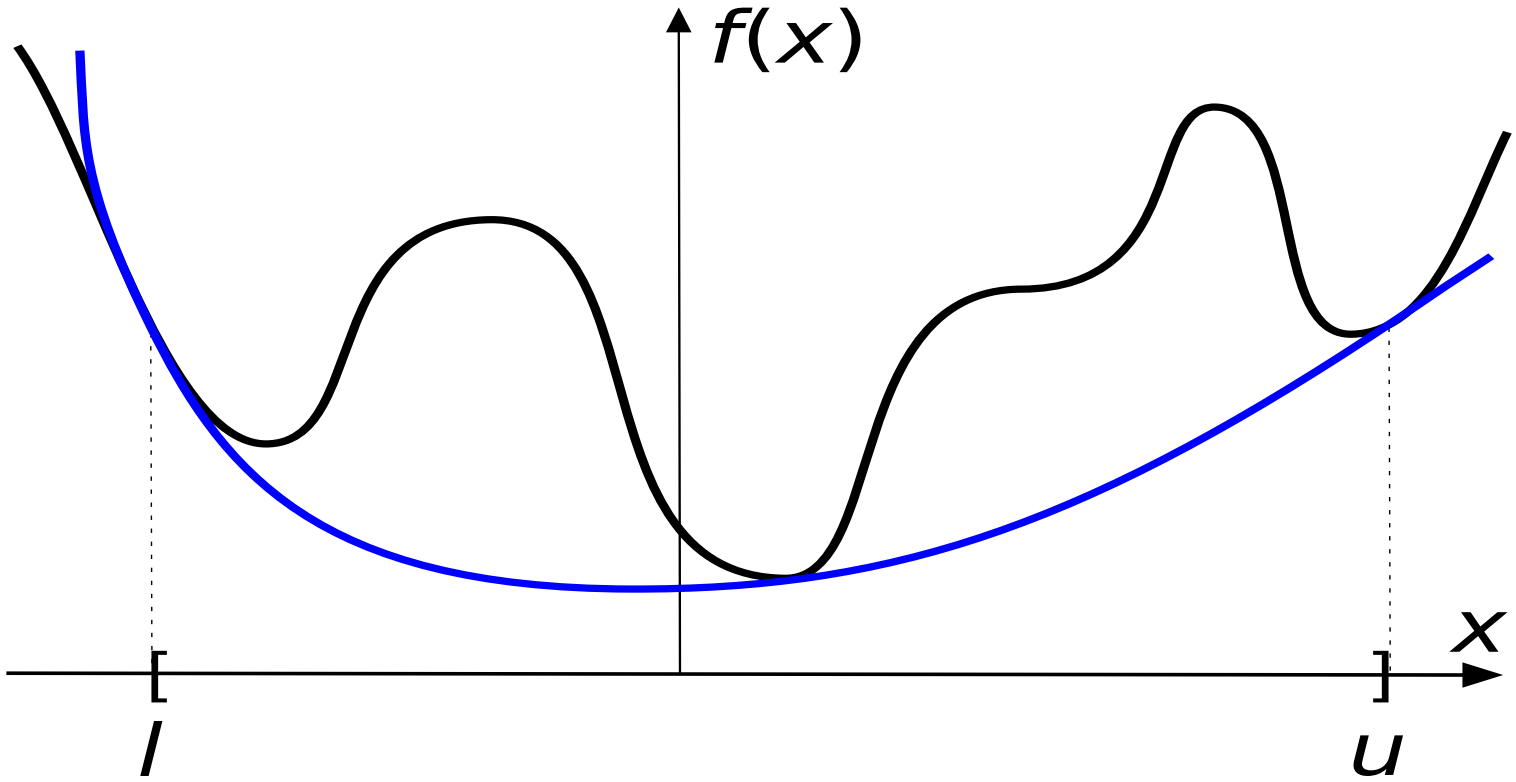
What Can I Do in the Nonconvex Case?

- Sift through all X but using a clever guide



What Can I Do in the Nonconvex Case?

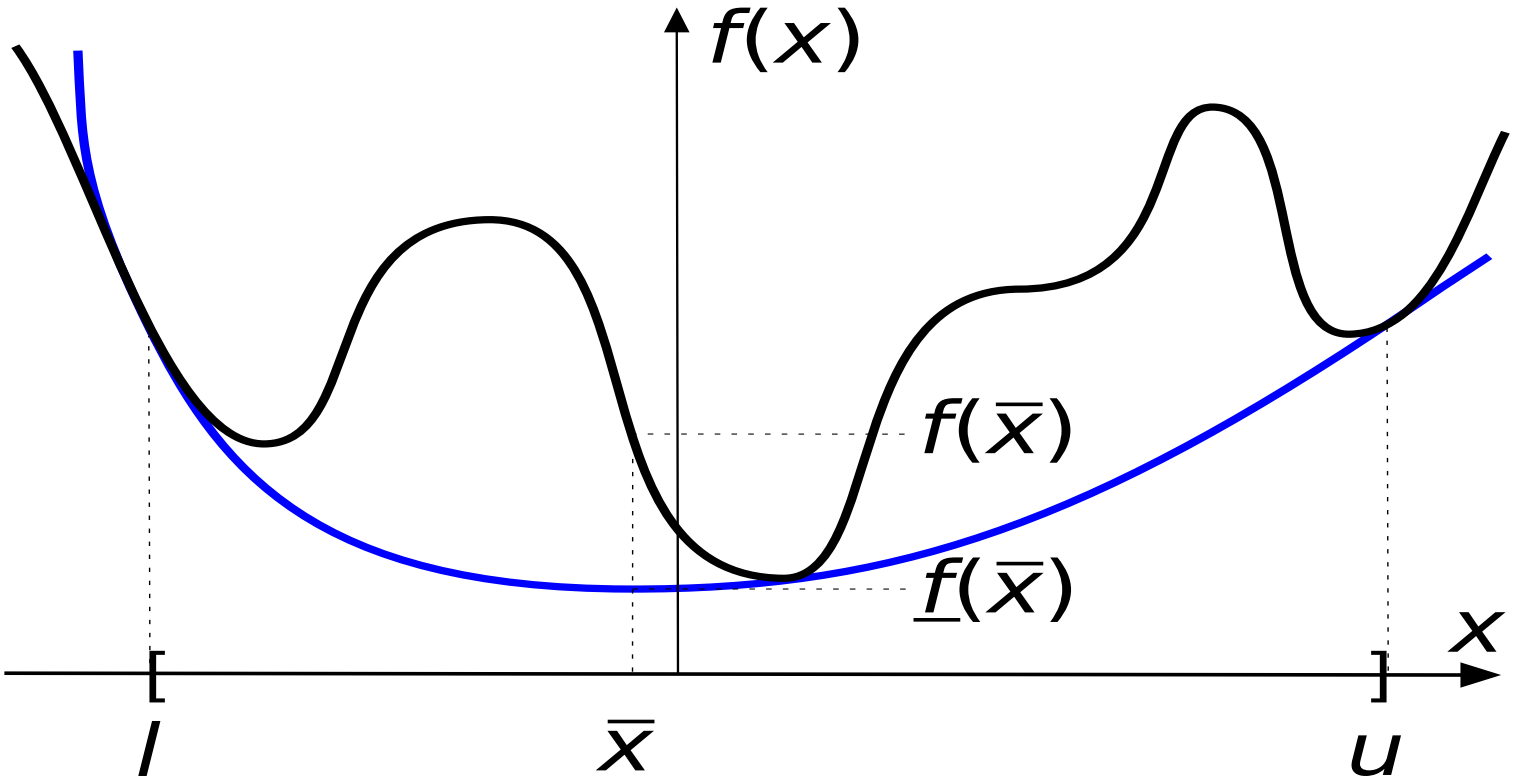
- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X

What Can I Do in the Nonconvex Case?

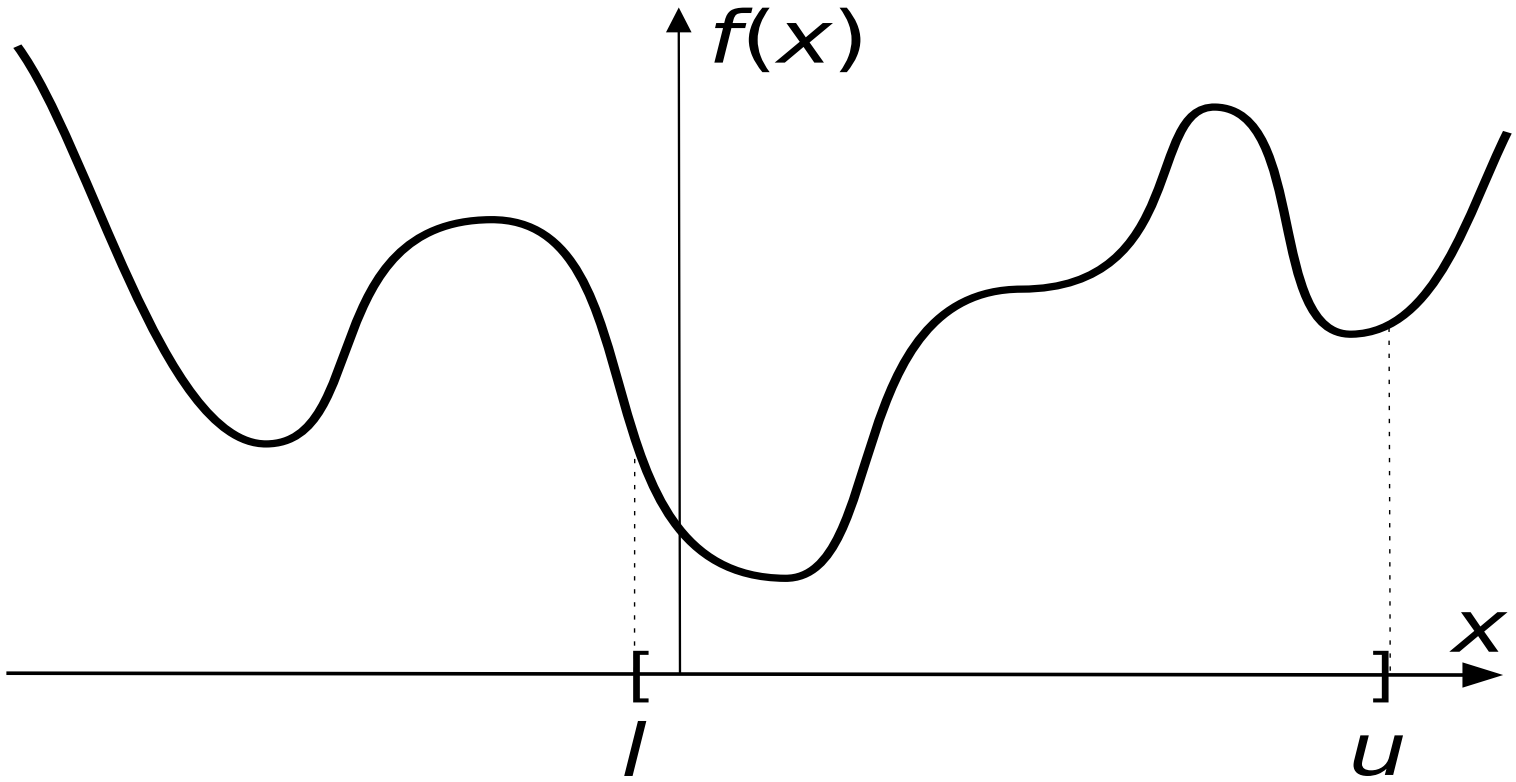
- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X
- Can easily find minimum $\bar{x}$, giving $\underline{f}(\bar{x}) \leq f_* \leq f(\bar{x})$

What Can I Do in the Nonconvex Case?

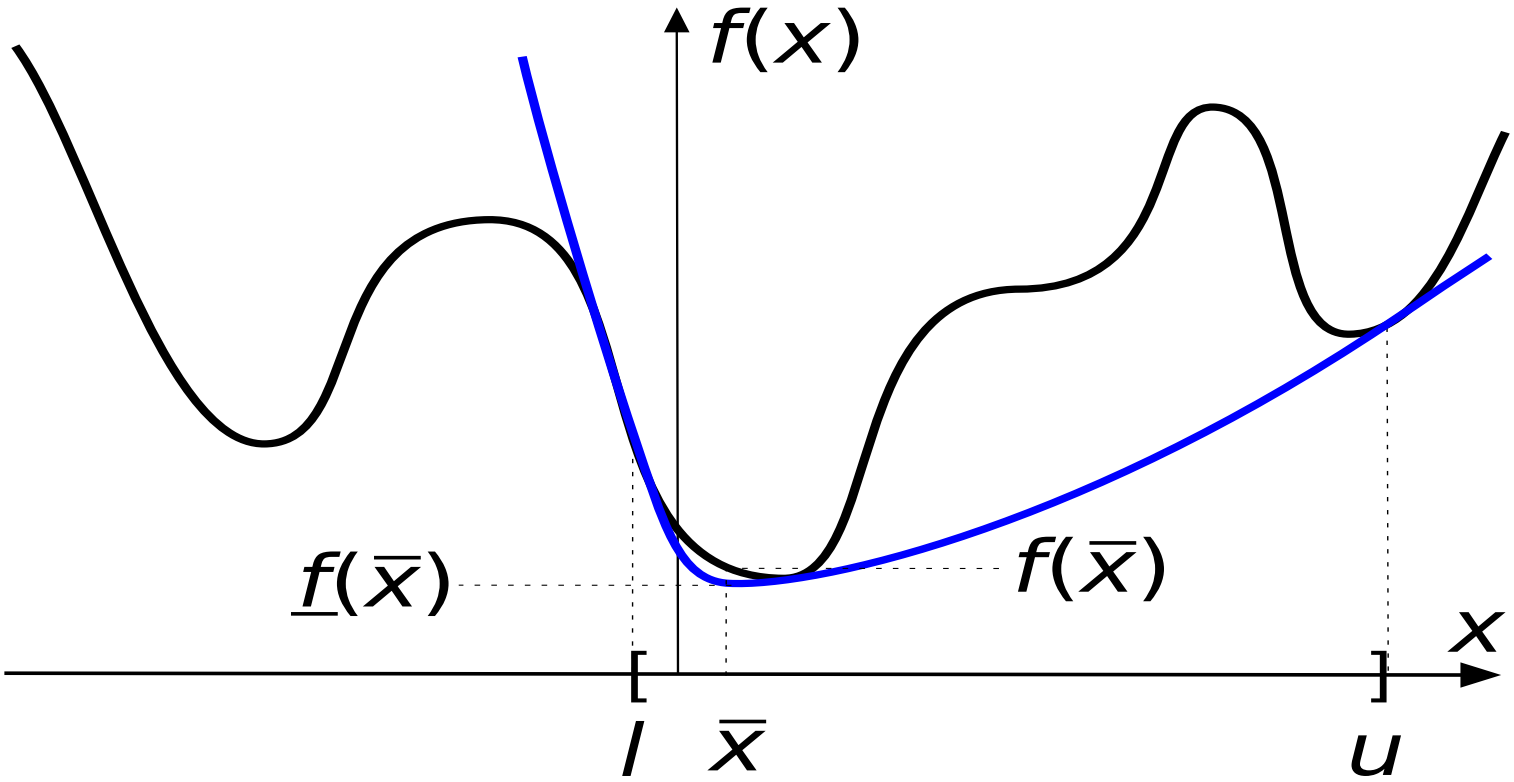
- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X
- Can easily find minimum $\bar{x}$, giving $\underline{f}(\bar{x}) \leq f_* \leq f(\bar{x})$
- If gap $f(\bar{x}) - \underline{f}(\bar{x})$ too large, partition X and iterate

What Can I Do in the Nonconvex Case?

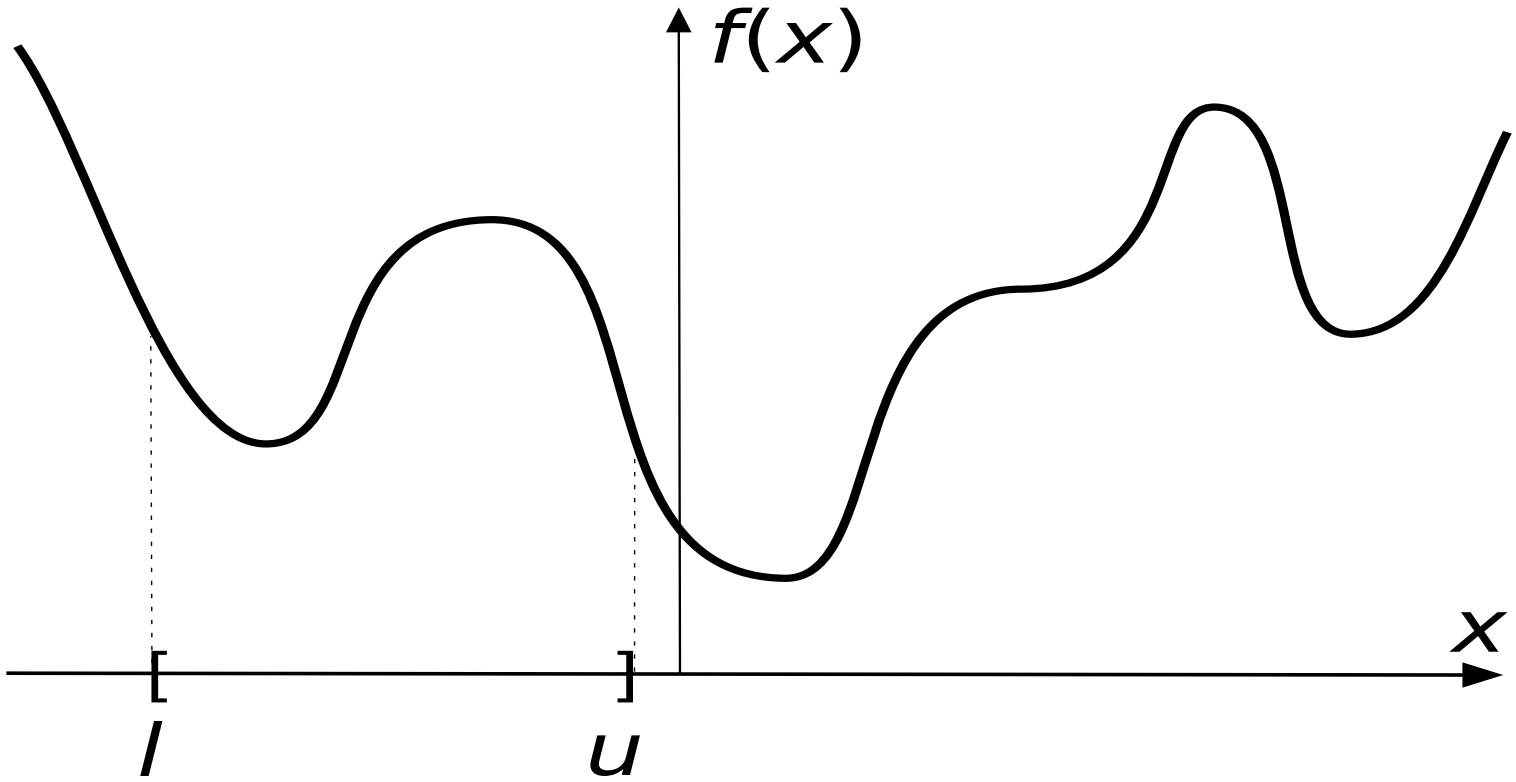
- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X
- Can easily find minimum $\bar{x}$, giving $\underline{f}(\bar{x}) \leq f_* \leq f(\bar{x})$
- If gap $f(\bar{x}) - \underline{f}(\bar{x})$ too large, partition X and iterate
- $\underline{f}$ depends on partition, smaller partition $\Longrightarrow$ better gap (hopefully)

What Can I Do in the Nonconvex Case?

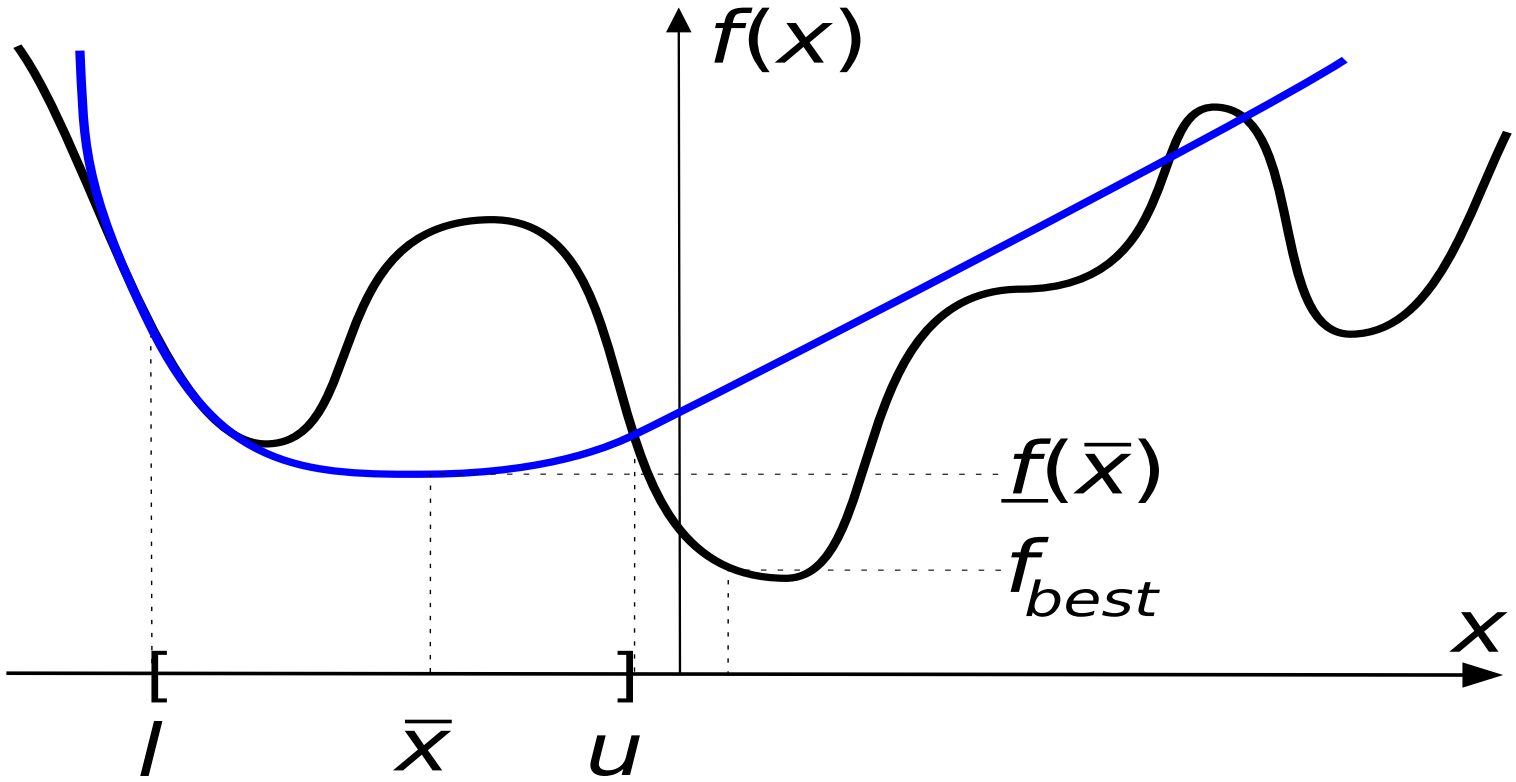
- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X
- Can easily find minimum $\bar{x}$, giving $\underline{f}(\bar{x}) \leq f_* \leq f(\bar{x})$
- If gap $f(\bar{x}) - \underline{f}(\bar{x})$ too large, partition X and iterate
- $\underline{f}$ depends on partition, smaller partition $\Longrightarrow$ better gap (hopefully)

What Can I Do in the Nonconvex Case?

- Sift through all X but using a clever guide



- Main idea: convex lower approximation $\underline{f}$ of nonconvex f on X
- Can easily find minimum $\bar{x}$, giving $\underline{f}(\bar{x}) \leq f_* \leq f(\bar{x})$
- If gap $f(\bar{x}) - \underline{f}(\bar{x})$ too large, partition X and iterate
- $\underline{f}$ depends on partition, smaller partition $\Longrightarrow$ better gap (hopefully)

Is Something Like This Efficient?

- In a word? Not really, but still feasible for not-too-large size
- Worst-case: keep dicing and slicing until pieces "very small"
- However, in practice it depends on:
  - "how much nonconvex" f really is
  - how good $\underline{f}$ is as a lower approximation of f
- Best possible lower approximation: convex envelope
- Bad news: computing convex envelopes is hard
- Typical approach:
  - rewrite the expression of f in terms of unary/binary functions
  - apply specific convexification formulæ for each component function

  $\Longrightarrow$ not the convex envelope
- Tedious job, bounds often rather weak
- Good news: implemented in available, well-engineered solvers
- It would help if finding the convex envelope were easy

Outline

A Very Convenient Form of Nonconvexity

- Computing "good" convex approximation both complex and difficult
- One very relevant case in which at least "easy": integrality constraints
- $x_i \in \mathbb{Z}$, most often $x_i \in \{ 0 , 1 \}$ very convenient for discrete choices (start machine/don't, make trip/don't, ...)
- Clearly nonconvex, $\exists$ nonlinear versions ($x_i(1 - x_i) \leq 0$, ...)
- Actually quite powerful: many different nonlinear nonconvex structures can be expressed via that + "simple" (linear) constraints
- Yet, this requires some rather weird formulation tricks
  $z = xy \equiv [(z \leq x) \wedge (z \leq y) \wedge (z \geq x + y - 1)]$ if all $x, y, z \in \{ 0 , 1 \}$
- If all the rest in the problem convex, then a convex relaxation very easy: continuous relaxation ($x_i \in \mathbb{Z} \longrightarrow x_i \in \mathbb{R}$)
- This does not mean convex relaxation is good, but it may be
- At least makes life a lot easier to solution algorithms

Going All the Way to Help The Solver

- Finding good relaxations crucial for practical efficiency
- Solvers helped a lot by having few, well-controlled nonconvexities
- Mixed-Integer Convex Problems the easiest class of hard problems
- Especially famous special case: Mixed-Integer Linear Program

  $$(MILP) \quad \min \left\{ cx \,:\, Ax \geq b \;,\; x_i \in \mathbb{Z} \;\; i \in I \right\}$$

  $\Longrightarrow$ continuous relaxation $\equiv$ Linear Program
- Very stable and efficient algorithms, some $\approx$ unique (simplex methods)
- Very powerful methods to improve relaxation quality (valid inequalities)
- Countless many results about special combinatorial structures (paths, trees, cuts, matchings, cliques, covers, knapsacks, ...)
- Clever approaches to exploit structure, though some work for MINLP too (Column/Row Generation, Dantzig-Wolfe/Benders' Decomposition, ...)

Put The Human in the Loop

- Fundamental point: formulating the problem well is crucial
- Almost anything can be written as a MILP, albeit to some $\approx$
  (not always a good idea: some nonlinearities "nice")
- Many different ways to write the same problem:
  apparently minor changes can make orders-of-magnitude difference
- Several of the best formulation weird and/or very large
  (appropriate tricks to only generate the strictly required part)
- Doing it "by hand" should not be required: solvers should
  be able to automatically find the best formulation (reformulate)
- Good news: the "perfect" formulation provably exists
- Bad news: it is provably ($\mathcal{NP}$-)hard to construct
- Doing it automatically is clearly difficult (but we should try harder)
- Meanwhile, a well-trained eye can make a lot of difference

Outline

An Instructive Test Case: p-slot Pickup & Delivery VRP

- Application that might arise in a shared economy setting: servicing customers with a fleet of vehicles with up to p "slots"
- Think servicing/moving bikes from stations in bike sharing
- Not really the application we studied it for, but:
  - typical: (at least part) of the problem has been studied before
  - hopefully instructive results
- Single depot d, heterogeneous fleet: k_s vehicles with $s = 1, \ldots, p$ slots
- Short time horizon: fixed customers, no bound on route length/duration
- Same vehicle moves $d \to i \in I$ (delivery) and $e \in E \to d$ (pickup), obvious logical constraint for slot occupancy, no $e \to i$ move
- Minimize cost/time/CO_2 emission, separable over individual legs
- Many (but not all) assumptions relatively easy to relax somewhat

Let's Play the Modelling Game

- "Abstract" directed graph $G = (\, d \cup I \cup E \,,\, A \,)$ of "direct transfers"
- Demand d_l for $l \in I \cup E$
- Not the "physical" graph:
  - $(\, i \,,\, j \,) \in A$ may "pass" some h but not stop
  - c_{ij} cost of shortest path (independent from load), $\neq c_{ji}$ (one-way streets)
  - same physical location can be both $i \in I$ and $e \in E$
- Fleet K of vehicles, each $k \in K$ with capacity $u_k \in \{\, 1 \,,\, \ldots \,,\, p \,\}$
- Have to invent appropriate variables:
  - $x_{ij}^{k} = 1$ if vehicle k traverses (just once) arc $(\, i \,,\, j \,)$
  - $y_{ij}^{k+} = \#$ of loaded slots for vehicle k on arc $(\, i \,,\, j \,)$ inbound to some I
  - $y_{ij}^{k-} = \#$ of loaded slots for vehicle k on arc $(\, i \,,\, j \,)$ outbound from some E
- Why those? Because flow structure well-known, nice properties

The Complete MILP Model

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k$$

$$s.t. \sum_{k \in K} \sum_{l \in N} y_{il}^{k\pm} = \sum_{k \in K} \sum_{j \in d \cup I} y_{ji}^{k\pm} \mp d_i \qquad i \in I$$

$$\sum_{k \in K} \sum_{l \in d \cup E} y_{el}^{k\pm} = \sum_{k \in K} \sum_{j \in N} y_{j}^{k\pm} \pm d_e \qquad e \in E$$

$$\sum_{l \in N} y_{il}^{k+} \leq \sum_{j \in d \cup I} y_{ji}^{k+}$$
$$\sum_{l \in N} y_{il}^{k-} \geq \sum_{j \in d \cup I} y_{ji}^{k-} \qquad i \in I \ , \ k \in K$$

$$\sum_{l \in d \cup E} y_{el}^{k+} \geq \sum_{j \in N} y_{je}^{k+}$$
$$\sum_{l \in d \cup E} y_{el}^{k-} \leq \sum_{j \in N} y_{je}^{k-} \qquad e \in E \ , \ k \in K$$

$$\sum_{(ji) \in A} (y_{ji}^{k+} + y_{ji}^{k-}) = \sum_{(il) \in A} (y_{il}^{k+} + y_{il}^{k-}) \qquad i \in I \cup E \ , \ k \in K$$

$$y_{ij}^{k+} + y_{ij}^{k-} \leq u_k x_{ij}^k \qquad (i,j) \in A \ , \ k \in K$$

$$\sum_{j \in N} x_{ji}^k - \sum_{l \in N} x_{il}^k = 0 \qquad i \in N \ , \ k \in K$$

$$\sum_{j \in N} x_{ij}^k \leq 1 \qquad i \in N \ , \ k \in K$$

$$x_{ij}^k \in \{0,1\} \ , \ y_{ij}^k \in \mathbb{N} \ , \ z_{ij}^k \in \mathbb{N} \qquad (i,j) \in A \ , \ k \in K$$

Er . . . What??

- Lots of nontrivial modelling tricks
- Independent flows for each vehicle, as well as outbound/inbound slots
- Logical constraints: $x_{ij}^k = 0 \Longrightarrow y_{ij}^{k+} = y_{ij}^{k-} = 0$
  (inherently exploiting nonconvexity of integer variables)
  also doubling up as capacity constraints
- Nontrivial logical relations: subtour elimination
- Not exactly a walk in the park: why the bother?
- Because lots of efficient software then available to (try to) solve it

The Software Tools

- Several steps: construct the instance, solve it, debug the model
- First already nontrivial, but lots of useful supporting software:
  - algebraic modelling languages, commercial[1] and open-source ones[2]
  - modelling systems in C++[3], Python[4], Julia[5], Matlab[6], ...
- General-purpose, well-engineered MILP solvers, commercial[7] and open-source ones[8], direct API access (C++, Java, Python, Matlab, ...)
- Project management tools (IDEs, database/spreadsheet interfaces, ...)
- Is it enough? Unfortunately in general it cannot be

---

[1] AMPL https://ampl.com — GAMS https://www.gams.com

[2] Coliop http://www.coliop.org — ZIMPL http://zimpl.zib.de

[3] FlopC++ https://projects.coin-or.orgFlopC++ — COIN-Reharse https://github.com/coin-or/Rehearse

[4] PuLP https://pythonhosted.org/PuLP — Pyomo http://www.pyomo.org

[5] JuMP https://github.com/JuliaOpt/JuMP.jl

[6] YALMIP https://yalmip.github.io

[7] Cplex https://www.ibm.com/analytics/cplex-optimizer — GuRoBi http://www.gurobi.com
MOSEK https://www.mosek.com

[8] Cbc https://projects.coin-or.org/Cbc — SCIP http://scip.zib.de

Some Results

- $p = 2$, 1h time, state-of-the-are general-purpose solver

$\|I\|$	$\|E\|$	k_1	k_2	time	gap
2	8	2	9	1920	0.00
5	5	2	7	980	0.00
2	8	0	10	–	3.76
8	2	0	12	–	3.83
2	28	13	33	–	7.54
5	25	0	36	–	11.4
28	2	0	42	–	3.79
28	2	14	35	–	–
20	20	0	35	–	–
25	15	0	43	–	18.9
10	40	18	50	–	–

- Gap to optimal solution, algorithmic gap much higher
- "Bad gap" = "bad relaxation" = "bad formulation" $\Longrightarrow$ no chance

Structure + Reformulation to the Rescue

- In our application, $p = 2$: structure to exploit
- $O(|I| \cdot |E|)$ possible routes ($d \to d$ cycles) for 1-slot vehicles and $O((|I| \cdot |E|)^2)$ for 2-slot ones
- Entire set of routes $R = R(1) \cup R(2)$ can be pre-computed
- Set partitioning reformulation

$$
\begin{aligned}
\min \ & \sum_{r \in R} c_r x_r & & \\
s.t. \ & \sum_{r \in R} \alpha_{I,r} x_r \geq d_I & & I \in I \cup E \\
& \sum_{r \in R(h)} x_r \leq k_h & & h = 1, 2 \\
& x_r \in \mathbb{N} & & r \in R
\end{aligned}
$$

($\alpha_{I,r} = \#$ of slots used for customer I in route r)

- Much simpler (complexity "hidden" in route construction)

More Results

- Same solver and machine as before, larger instances

				MILP	*SC*		
$\|I\|$	$\|E\|$	k_1	k_2	gap	$\|R\|$	tM	time
20	5	4	56	–	1.1e+4	1.0	0.2
20	20	4	47	–	1.7e+5	9.0	1.0
30	15	6	69	–	2.1e+5	6.8	13.3
45	12	8	112	–	3.0e+5	8.7	16.4
45	45	9	129	–	4.2e+6	30.5	782.3
75	38	13	177	–	8.2e+6	49.3	–
100	25	17	236	–	6.3e+6	38.1	–

- Can easily solve much larger instances
- But size growing quartically takes a toll eventually

Dynamically Generating Large Formulations

- Issue: formulation is large, but clearly only "few" routes actually used
- If only I knew which are the "good" routes (but I don't)
- Said otherwise: given $R' \subset R$, what is the route to add that helps most?
- For the continuous relaxation this can be answered
- Dual of the (continuous relaxation) of Set-Partitioning formulation

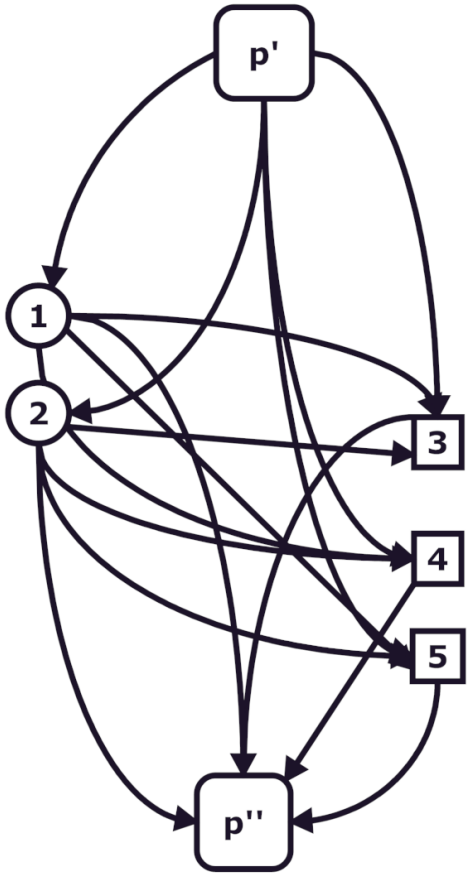
$$
\begin{aligned}
\max \quad & \textstyle\sum_{l \in I \cup E} \xi_l d_l - \sum_{h=1,2} \pi_h k_h & & \\
s.t. \quad & \textstyle\sum_{l \in I \cup E} \xi_l \alpha_{l,r} - \pi_{h(r)} \leq c_r & & r \in R \\
& \xi_l \in \mathbb{R}_+ & & l \in I \cup E \\
& \pi_h \in \mathbb{R}_+ & & h = 1, 2
\end{aligned}
$$

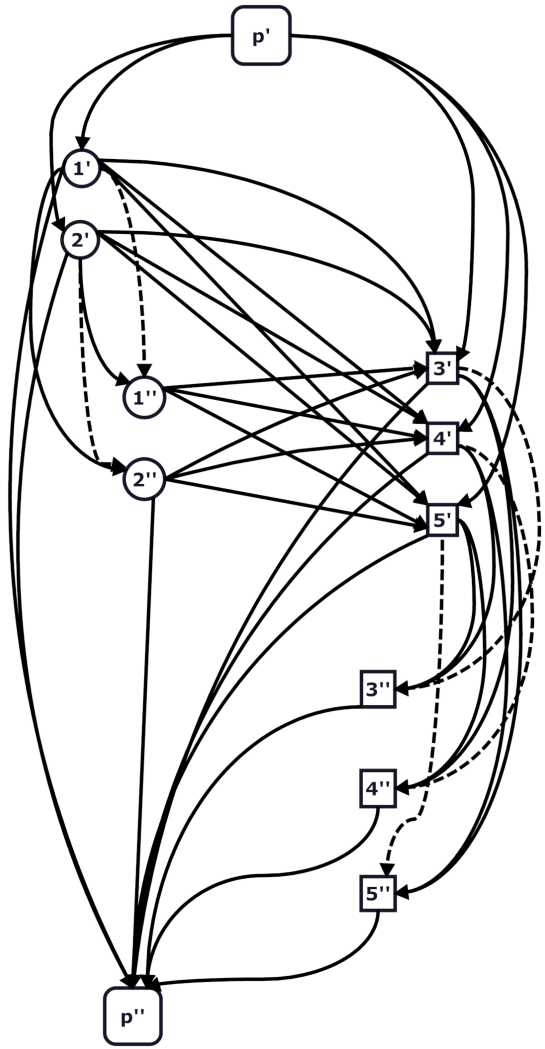
($h(r)$ = what kind of vehicle route r is of)

- Few variables but many constraints

Column Generation = Separation in the Dual

- Given (ξ^*, π^*), route *r* helps $\iff$ corresponding constraint violated
- Can be written as a shortest path on a carefully tailored graph
- Only solves continuous relaxation, but gives promising set of routes
- P&B = find best solution only using these
- Solve a MILP, but a small one
- Heuristic, but fast





Even More Results

- Same solver and machine as before, larger instances

				SC			*P&B*	
$\|I\|$	$\|E\|$	k_1	k_2	$\|R\|$	tM	time	time	gap
20	5	4	56	1.1e+4	1.0	0.2	1.1	3.04
20	20	4	47	1.7e+5	9.0	1.0	3.0	5.08
30	15	6	69	2.1e+5	6.8	13.3	3.5	4.11
45	12	8	112	3.0e+5	8.7	16.4	3.4	4.61
45	45	9	129	4.2e+6	30.5	782.3	11.7	5.18
75	38	13	177	8.2e+6	49.3	–	16.2	4.79
100	25	17	236	6.3e+6	38.1	–	15.3	4.79

- Very fast, always finds a solution
- Solutions not always very good
- Could embed this in enumeration (B&P), optimal solution but slower

Yet Another Reformulation

- Crucial point: shortest paths are a flow problem on $\neq$ graphs

  $\bar{G}_h = (\bar{N}_h, \bar{A}_h)$, $\{v', v''\} \subset \bar{N}_h$ for $v \in I \cup E$, d splitted in d', d''

- "Just write that problem" inside the Set-Partitioning formulation

$$\min \sum_{h=1,2} \sum_{(i,j)\in\bar{A}_h} c_{ij} x^h_{ij}$$

$$s.t. \sum_{(j,i)\in BS_h(i)} x^h_{ji} - \sum_{(i,j)\in FS_h(i)} x^h_{ij} = 0 \qquad i \in \bar{N}_h \;,\; h = 1,2$$

$$\sum_{h=1,2} \left[\sum_{(j,v)\in BS^1_h(v)} x^1_{jv} + \sum_{(j,v')\in BS^2_h(v')} x^2_{jv'} + \sum_{(j,v'')\in BS^2_h(v'')} x^2_{jv''} \right] \geq d_v \quad v \in I \cup E$$

$$x^h_{d''d'} \leq k_h \qquad h = 1,2$$

$$x^h_{ij} \in \mathbb{N} \qquad (i,j) \in \bar{A}_h \;,\; h = 1,2$$

- Strikingly similar to first one: some flows + other constraints
- Performance-wise could not be more $\neq$

Yet More Results

- Yet again same solver and machine as before, "large" (??) instances

				SC			*P&B*		SEAF
$\|I\|$	$\|E\|$	k_1	k_2	$\|R\|$	tM	time	time	gap	time
20	5	4	56	1.1e+4	1.0	0.2	1.1	3.04	0.71
20	20	4	47	1.7e+5	9.0	1.0	3.0	5.08	1.21
30	15	6	69	2.1e+5	6.8	13.3	3.5	4.11	0.63
45	12	8	112	3.0e+5	8.7	16.4	3.4	4.61	0.44
45	45	9	129	4.2e+6	30.5	782.3	11.7	5.18	2.02
75	38	13	177	8.2e+6	49.3	–	16.2	4.79	3.47
100	25	17	236	6.3e+6	38.1	–	15.3	4.79	3.64

- From "no solution in an hour" to "optimal solution in < 4 seconds"
- The only thing that changes is the formulation
- Can be extended to other cases (e.g., multi-period[9])

[9] Ghezelsoflu, Di Francesco, F., Zuddas "A Multiperiod Drayage Problem with Customer-dependent Service Periods", 2019

Conclusions

- Optimization problems are difficult, but there are $\neq$ kinds of "difficult"
- First crucial choice: which class of optimization problems
- MILPs should be the first choice (actually MI-SOCPs):
  lots of stable, well-developed software, even open-source
- Many problems have structure that can be exploited
- Apparently very complex problems may not be that difficult
  if one knows the right set of modelling tricks
- Some mad people having fun just with solving optimization problems:
  though applications need good methodologies, but
  good methodologies learn a lot from though applications
- Optimization a minority stake in any good application, but
  at the right moment it can save your day, as in sharing economy

An Incredibly Nifty Trick: (Mixed-Integer) Conic Programs

- Good news: can "hide" many nonlinearities in a "Linear" Program
- Conic Program: $(P)\ \ \min\{\, cx \,:\, Ax \geq_K b \,\}$

  where $x \geq_K y \equiv x - y \in K$, K pointed convex cone, e.g.
  - $K = \mathbb{R}^n_+ \equiv$ sign constraints $\equiv$ Linear Program
  - $K = \mathbb{L} = \left\{ x \in \mathbb{R}^n \,:\, x_n \geq \sqrt{\sum_{i=1}^{n-1} x_i^2} \right\} \equiv$ Second-Order Cone Program
  - $K = \mathbb{S}_+ = \{\, A \succeq 0 \,\} \equiv$ "$\succeq$" constraints $\equiv$ SemiDefinite Program
- Exceedingly smart idea: everything is linear, but the cone is not

  $\equiv$ a nonlinear program disguised as a linear one
- Many interesting (convex) nonlinear functions are conic-representable

  (but have to learn some even weirder formulation trick)
- Continuous relaxation almost as efficient as Linear Program
- Many combinatorial MILP tricks extend di MI-SOCP (valid surfaces, . . .)
- Support in general-purpose software growing, already quite advanced