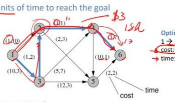
Constrained Shortest Path Problem

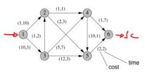
- Let's add the following constraint:
  - We cannot exceed 14 units of time to reach the goal



Optimal solution:
1 → 3 → 2 → 4 → 6
cost: 13
time: 13

- Can you use A* to solve this problem?
  - Not as it is. We would need to change the **problem representation** to distinguish reaching states at different times, e.g., reaching state 2 using
  - 1 unit of cost and 10 units of time (1 → 2)
  - 11 units of cost and 5 units of time (1 → 3 → 2) ← Should you prune this option?
- Complexity: Constrained SPs are $\mathcal{NP}$-hard

Constrained SP as an ILP

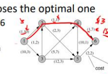


$$z^* := \min \sum_{(i,j)\in A} c_{ij}x_{ij}$$
$$\text{s.t.} \sum_{j:(1,j)\in A} x_{1j} = 1$$
$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = 0 \quad i = 2,3,4,5$$
$$\sum_{i:(i,6)\in A} x_{i6} = 1$$
$$\sum_{(i,j)\in A} t_{ij}x_{ij} \le 14$$
$$x_{ij} \in \{0,1\} \quad (i,j)\in A$$

- Flow model
  - push 1 unit of flow into node 1
  - preservation of flow
  - extract 1 unit of flow from node 6
- Each arc (i,j) ∈ A is either used or not
- We can solve with Branch and Bound

Path Reformulation

- Let's enumerate all paths and write an ILP that chooses the optimal one
  - y_{1246} is a binary variable representing the path 1→2→4→6
  - Object function coefficient is the cost of the path
  - Coefficient in time constraint is the time used by this path



Variables (Columns) used by Column Generation

$$\begin{array}{llllllllll} \min & 3y_{1246} + & 14y_{12456} + & 5y_{1256} + & 13y_{13246} + & 24y_{132456} + & 15y_{13256} + & 16y_{1346} + & 27y_{13456} + & 24y_{1356} \\ \text{s.t.} & 18y_{1246} + & 14y_{12456} + & 15y_{1256} + & 13y_{13246} + & 9y_{132456} + & 10y_{13256} + & 17y_{1346} + & 13y_{13456} + & 8y_{1356} \le 14 \\ & y_{1246} + & y_{12456} + & y_{1256} + & y_{13246} + & y_{132456} + & y_{13256} + & y_{1346} + & y_{13456} + & y_{1356} = 1 \\ & y_{1246}, & y_{12456}, & y_{1256}, & y_{13246}, & y_{132456}, & y_{13256}, & y_{1346}, & y_{13456}, & y_{1356} \in \{0,1\} \end{array}$$

- This ILP is correct and will find the opt. solution. What is the issue with it?
  - It has a lot of variables!
- But do we need all the path variables? No

Path Reformulation ILP

$$z^* = \min \sum_{p\in P} \left(\sum_{(i,j)\in A} c_{ij}x_{pij} \right) y_p$$ ← Inner summation is the total time consume by path p

$$\text{s.t.} \sum_{p\in P} \left(\sum_{(i,j)\in A} t_{ij}x_{pij} \right) y_p \le 14$$

$$\sum_{p\in P} y_p = 1$$ ← We want a convex combination of paths. Together with x_{ij} ∈ {0,1}, only one path will be selected

$$y_p \ge 0 \quad p\in P$$ ← Relax the path variables to be continuous

$$\sum_{p\in P} x_{pij}y_p = x_{ij} \quad (i,j)\in A$$ ← x_{pij} is a **binary** constant and it is 1 if i→j in path p. This constraints links x_{ij} and y_p

$$x_{ij} \in \{0,1\} \quad (i,j)\in A$$ ← Each arc can be used at most once

- Column Generation idea: the constraint matrix is so large (and implicit) that we will generate it on demand
  - In the Constraint SP, we will generate the paths on demand

Column Generation: Road Map

- We have two optimization problems:
  - **Reduced Master Problem** (RMP): original problem with just a subset of variables
  - **Pricing problem**: the problem that will generate new variables (columns) for the RMP
  - The original reformulated problem is referred as the **Master Problem**
- For the Linear Relaxation of Constrained Shortest Path:

Reduced Master Problem (RMP)

$$z^* = \min \sum_{p\in P'} \left(\sum_{(i,j)\in A} c_{ij}x_{pij} \right) y_p$$
$$\text{s.t.} \sum_{p\in P'} \left(\sum_{(i,j)\in A} t_{ij}x_{pij} \right) y_p \le 14$$
$$\sum_{p\in P'} y_p = 1$$
$$y_p \ge 0 \quad p \in P' \subseteq P$$

Pricing Problem

???

???
New variable y_i
???
New variable y_j
???
No good variable left

Pricing Problem

- Goal: find a variable to be added to the LP to improve its objective function
- Does it sound familiar? Revised Simplex

$$\min_{x\in\mathbb{R}^n} c^\intercal x \quad \text{s.t. } Ax = b, \quad x \ge 0$$

Basic Variables: $x_j \ge 0 \quad \forall j \in B_k$ $\qquad \bar{A}\bar{x} = b$

Non-Basic Variables: $x_j = 0 \quad \forall j \notin B_k$ $\qquad \tilde{x} = 0$

Reduced Cost: the marginal increase in objective for a marginal increase in each non-basic variable

Pick entering variable: (otherwise optimal) $\bar{A}^\intercal\lambda = \bar{c} \Rightarrow \mu = \tilde{c} - \tilde{A}^\intercal\lambda \Rightarrow u \in \{u | \mu_u < 0\}$

Dual variables for the **basic variables**

BAR ← BASIC
NON-B. PART OF A
OBJ FUNC COEF FOR NB VARS

Column Generation: Algorithm

1. Start with some columns
2. Solve the problem using those columns (**Restricted Master Problem**)
3. Use the dual variables to define a new subproblem (**Pricing Problem**)
4. Solve the pricing problem to obtain the column with **minimum reduced cost**
5. If the reduced cost of the new column is negative, add it to the RMP and go to 2. Otherwise, we have the optimal solution to the linear relaxation of the problem

Constrained SP: Linear Relax. Full Example

RMP:

$$\begin{array}{lll} \min & 3y_{1246} + 24y_{1356} & \\ \text{s.t.} & 18y_{1246} + 8y_{1356} & \le 14 \ (\lambda_1) \\ & y_{1246} + y_{1356} & = 1 \ (\lambda_0) \\ & y_{1246}, \ y_{1356} & \ge 0 \end{array}$$

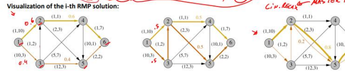
OPT OF ILP 1→3→2→4→6

RMP solution $\quad \bar{z} \quad \lambda_0 \quad \lambda_1 \quad$ Pricing Problem $\min_p \mu_p \quad p \quad c_p \quad t_p$

SOL. FOR MASTER PROB

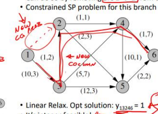
LIN. RELAX

Visualization of the i-th RMP solution:



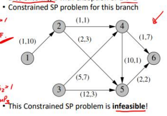
Constrained SP: Branching on x_{12}

Branch x_{12} == 0
- Remove all y variables that use 1→2, e.g., y_{1246}, y_{1256}, etc.
- Can do so by removing arc 1 → 2 from the graph
- Constrained SP problem for this branch



- Linear Relax. Opt solution: y_{13246} = 1
- It's integer feasible!

Branch x_{12} == 1
- Remove all y variables that **do not use** 1→2
- Can do so by removing all arcs leaving 1 and all arcs entering 2 with exception of 1→2
- Constrained SP problem for this branch

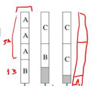


- This Constrained SP problem is **infeasible**!

This approach (B&B + CG) is known as branch-and-price

Cutting Stock: Reformulation (1)

- A *pattern* is a list of the products we will cut from a rod
  - E.g. 3 of product A and 1 of B, or 1 of product C and 1 of B
- We ensure feasibility *a priori*
  - Don't allow the pattern if the sum of lengths exceeds maximum



rows: offered lengths
columns: patterns
a_{ij}: number of items of length i produced by pattern j

A	3	0	0	2
B	1	1	0	0
C	0	1	2	0
D	0	0	0	2
E	0	0	0	1

Cutting Stock: Reformulation (2)

Let's say we can list *all possible* patterns

- Now our formulation is easy
- x_j is: how many times do we use pattern *j*?

A	3	0	0	2	...
B	1	1	0	0	...
C	0	1	2	0	...
D	0	0	0	2	...
E	0	0	0	1	...

$$\min \sum_j x_j \quad \text{s.t.} \sum_j a_{ij}x_j = d_i \quad \forall i, \quad x_j \in \mathbb{N}_0$$

$$\Rightarrow \min \sum_j x_j \quad \text{s.t.} \sum_j a_{ij}x_j \ge d_i \quad \forall i, \quad x_j \in \mathbb{N}_0$$

SUPP NEED 1 A

make it easier by producing a little extra instead of having a pattern that produces a single length

Cutting Stock: Pricing Problem (1)

- Recall the reduced cost is: $\mu = \tilde{c} - \tilde{A}^\intercal\lambda$
- For cutting stock problem, the reduced cost for a pattern j is: $\mu_j = 1 - \sum_i a_{ij}\lambda_i$

RMP

$$\min \sum_j x_j \quad \text{s.t.} \sum_j a_{ij}x_j \ge d_i \quad \forall i \ (\lambda_i), \quad x_j \in \mathbb{N}_0$$

- We want to find a **feasible** pattern that minimises it:

$$\min_a 1 - \sum_i a_i\lambda_i \quad \text{s.t.} \sum_i w_ia_i \le W, \quad a_i \in \mathbb{Z}_{\ge 0}$$

$$= 1 - \max_a \sum_i a_i\lambda_i \quad \text{s.t.} \sum_i w_ia_i \le W, \quad a_i \in \mathbb{Z}_{\ge 0}$$

VALUE
WEIGHT

Cutting Stock: Pricing Problem (2)

$$\max_a \sum_i a_i\lambda_i \quad \text{s.t.} \sum_i w_ia_i \le W, \quad a_i \in \mathbb{Z}_{\ge 0}$$

ALSO NP-HARD

- This gives us a knapsack problem
  - Weight: the offered lengths w_i (pick lengths totalling up to W)
  - Value: the *reduced cost* derived from dual variables
- Yes, we have just replaced one $\mathcal{NP}$-hard problem with another, BUT
  - We only have one rod now, not *k* of them
  - Knapsack is an "easy" NP problem
    - The heuristic solution offers very good bounds on the optimal value

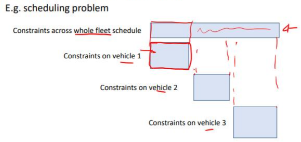
Cutting Stock: Column Generation

ROD: 100cm TYPE: A = 10cm B = 20cm C = 50cm

1. Start with some columns
   - Cutting stock: can you think about an easy set of patterns to start with?
2. Solve the **Restricted Master Problem**
   - Cutting stock: the problem with the patterns we have so far
3. Use the dual variables to define the **Pricing Problem**
   - Cutting stock: knapsack problem with values derived from the dual variables
4. Solve the pricing problem to obtain the column with **minimum reduced cost**
   - Cutting stock: solve the knapsack problem
5. If the reduced cost of the new column is negative, include it, and go to 2. Otherwise, we have the optimal solution to the relaxed (non-integer) problem
   - Cutting stock: if a pattern with negative reduced cost was found, add it to the RMP

Dantzig-Wolfe Decomposition

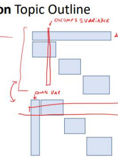
- Column Generation helps with complicating constraints
- Dantzig-Wolfe takes that to the next level
- E.g. scheduling problem



Decomposition Topic Outline

- Column Generation
- Bender's Decomposition
  - Facility Location problem
  - Derivation
  - Numerical Example
  - Connection with Column Generation
  - Real World Application: BusPlus

IN CANBERRA



Benders: Intuition

So for the *original problem* (**OP**)

OP: minimise $f^\intercal y + c^\intercal x$
subject to $Ay = b$
$By + Dx = d$
$x \ge 0$
$y \in \mathbb{Z}^+$

IF y IS FIXED
$f^\intercal \hat{y} + \min c^\intercal x$ s.t. $Dx = d - B\hat{y}$
CONS
MUST BE CONT.
COMPL.

Benders solves the problem in *x* for a fixed value of *y*

Benders: Derivation

OP: minimise $f^\intercal y + c^\intercal x$
subject to $Ay = b$
$By + Dx = d$
$x \ge 0$
$y \in \mathbb{Z}^+$

$\Rightarrow Y = \{y \mid Ay = b, y \in \mathbb{Z}^+\}$

LIST OF FEAS SCH TO BUILD

OP can be rewritten as

$$\min_{\hat{y}\in Y}\{f^\intercal\hat{y} + \min_{x\ge 0}\{c^\intercal x : Dx = d - B\hat{y}\}\}$$

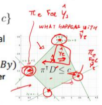
ŷ IS FIXED
SUB P
FIND MIN COMMUTE FOR FAM.
FEAS ŷ

Benders: Derivation – DSP is feasible

DSP: $\max_\pi\{\pi^\intercal(d - B\hat{y}) : \pi^\intercal D \le c\}$

π_e FOR $\hat{y}_1$
WHAT HAPPENS WITH $\hat{y}_2$

- If feasible, π_e that achieves the maximum is a potential solution
- The maximum of the DSP for any *y* is **at least** $\pi_e(d - By)$
  - Why? Because π_e is a feasible point and **the constraints never change**
- We can theoretically enumerate every extreme point (vertex) $\pi_e \in E$



$$\min_{\hat{y}\in Y}\{f^\intercal\hat{y} + \underbrace{\min_{x\ge 0}\{c^\intercal x : Dx = d - B\hat{y}\}}_{DSP = \eta}\} \Rightarrow \left[\begin{array}{l} \min_{y,\eta} f^\intercal y + \eta \\ \eta \ge \pi_e(d - By) \quad \forall \pi_e \in E \end{array} \right.$$

Optimality cuts

Benders: Derivation

Benders Master Problem

BMP:

$$\begin{array}{lll} \min_{y,\eta} & f^\intercal y + \eta & \\ \text{subject to} & Ay = b & \\ & \eta \ge \pi_e^\intercal(d - By) & \forall \pi_e \in E \quad \text{Optimality cuts} \\ & 0 \ge r_q^\intercal(d - By) & \forall r_q \in Q \quad \text{Feasibility cuts} \\ & y \in \mathbb{Z}^+ & \end{array}$$

EXPONENTIALLY LARGE SETS
GENERATE THEM ON DEMAND
CONSTR. GEN

Iteration 1a

- Let's start with $\hat{y} = 0$
- Solve subproblem $\max 2(0) + \pi^\intercal\left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}1\\3\end{bmatrix}(0)\right)$ subject to $\pi^\intercal\begin{bmatrix}1 & 2\\2 & -1\end{bmatrix} \le \begin{bmatrix}2\\3\end{bmatrix}$, $\pi \ge 0$
- The optimal solution is $\pi_e^\intercal$ = [1.6, 0.2], giving UB 5.6
- Add cut $\eta \ge \pi_e^\intercal(d - By)$

$$\eta \ge [1.6 \ 0.2]\left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}1\\3\end{bmatrix}y\right)$$
$$\eta \ge 5.6 - 2.2y$$

$f = [2]$, $c^\intercal = [2, 3]$, $B = \begin{bmatrix}1\\3\end{bmatrix}$, $D = \begin{bmatrix}1 & 2\\2 & -1\end{bmatrix}$, $d^\intercal = [3, 4]$

Iteration 1b

- Solve the reduced master problem

$$\min 2y + \eta \quad \text{s.t. } \eta \ge 5.6 - 2.2y, \quad y \ge 0$$

- Optimal solution $(\hat{y}, \eta) = (2.545, 0)$, $LB = 5.091$

Iteration 2a

- Solve the sub problem

$$\max 2(2.545) + \pi^\intercal\left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}1\\3\end{bmatrix}(2.545)\right) \quad \text{subject to } \pi^\intercal\begin{bmatrix}1 & 2\\2 & -1\end{bmatrix} \le \begin{bmatrix}2\\3\end{bmatrix}, \quad \pi \ge 0$$

- Optimal solution is $\pi_e^\intercal$ = [1.5, 0], giving UB 5.772
- Add cut $\eta \ge 4.5 - 1.5y$

Iteration 2b

- Solve the reduced master problem

$$\min 2y + \eta \quad \text{s.t. } \eta \ge 5.6 - 2.2y, \quad \eta \ge 4.5 - 1.5y, \quad y \ge 0$$

- Optimal solution $(\hat{y}, \eta) = (1.571, 2.143)$, $LB = 5.286$

Iteration 3a

- Solve the sub problem

$$\max 2(1.571) + \pi^\intercal\left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}1\\3\end{bmatrix}(1.571)\right) \quad \text{subject to } \pi^\intercal\begin{bmatrix}1 & 2\\2 & -1\end{bmatrix} \le \begin{bmatrix}2\\3\end{bmatrix}, \quad \pi \ge 0$$

- Optimal solution is $\pi_e^\intercal$ = [1.6, 0.2], giving UB 5.286
- LB == UB, so we can stop.
- Solution is (y=1.571, x_1=0, x_2=0.714), with value 5.286

Obtained using $Dx = d - B\hat{y}$

Benders Decomposition

LOGIC BENDERS DEC.

- No integer variables in the subproblem
  - We use the dual to solve the subproblem
  - The value of dual variables is not well-defined if we have integer constraints
  - The algorithm as presented cannot be used
- Subject to slow convergence,
- Many improvements, tricks and add-ons developed.
- For a recent and excellent review, see:

Rahmaniani, Ragheb & Crainic, Teodor Gabriel & Gendreau, Michel & Rei, Walter. (2017). **The Benders Decomposition Algorithm: A Literature Review**. *European Journal of Operational Research* (259) pp 801-817. doi:10.1016/j.ejor.2016.12.005

Relationship between CG and BD

- If both problems are all-continuous (no integer variables) then CG and BD are indeed duals
  - The complicating variables in the CG become the complicating constraints in BD
  - The subproblems are equivalent.
- However, in the presence of integer constraints, the relationship is complex.
  - BD directly converges to a solution of the MIP, not a relaxation
    - No additional B&B tree
  - BD can't handle integer constraints in the subproblem
    - CG is happy with that

BD | B&B$_1$ | B&B$_2$ | B&B$_3$ — CG: RMP, RMP$_2$, RMP$_3$

Convex Functions

We've introduced convex sets. We can also talk about the convexity of functions, which is useful for determining whether our objective and constraints represent a convex optimisation problem.

A real-valued **function is convex** if and only if the line segment between any two points on its surface is **on or above** its *graph*:

$f : D \to \mathbb{R}$ $\quad D \quad$ A convex set in some real vector space

$$\forall x, y \in D: \; f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \qquad \forall \alpha \in [0,1]$$

ABOVE FUNC; $\alpha=0$, $\alpha=0.5$, $\alpha=1$

strict convexity if a strict inequality for **open** line segment

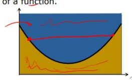
Conversely: a real-valued **function is concave** if and only if the line segment between any two points on its surface is **on or below** its *graph*.

Multi-variate and Vector-valued

A **convex** function's *epigraph* is a convex set.

A **concave** function's *hypograph* is a convex set.

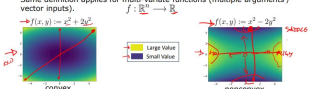
Epigraph (**hypograph**) = set of points on or **above** (**below**) the *graph* of a function.



Note that might only be interested in the convexity of a function over an interval or convex set (single or multi-variate) rather than the full space.

Same definition applies for multi-variate functions (multiple arguments / vector inputs). $\quad f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$f(x,y) := x^2 + 2y^2$ $\qquad$ $f(x,y) := x^2 - 2y^2$ SADDLE



convex $\qquad$ nonconvex

A convex vector-value function (vector output): each output is convex.

$$f : \mathbb{R} \longrightarrow \mathbb{R}^m \qquad f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix} \quad \text{all convex}$$

Combining Functions

$f(x) = x + x^2$ (BOTH CONVEX CONC)

Recall, intersection of convex sets is itself a convex set. Another important property:

The **sum of convex functions** results in **a convex function**.

$f(x) = x^2 + e^x - \log(x)$ $\quad$ is convex over the interval $x \in (0, \infty)$ because each term is convex

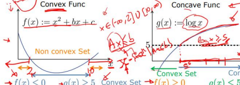
$f(x,y) = x^2 + y^2$ $\quad$ both terms are convex again

We can also multiply functions by a positive scalar without changing convexity.

Multiplying by a negative scalar swaps from convex to concave and vice versa.

Functions in Constraints

$f(x)$ is convex $\qquad$ $g(x)$ is conc



Convex Func: $f(x) := x^2 + bx + c$ $\qquad$ Concave Func: $g(x) := \log x$

Non convex Set $\qquad$ Convex Set $\qquad$ Convex Set $\qquad$ Convex Set

$x \in (-\infty, 2] \cup [10, \infty)$ $\quad$ $Ax \leq b$ $\quad$ $\log x \geq 5$ $\quad$ $x \in [e^5, \infty)$

$f(x) \leq 0$ $\quad$ $g(x) \geq 5$ $\quad$ Convex Set $\quad$ $f(x) \geq 0$ $\quad$ $g(x) \leq 5$

How do we interpret these? Feasible region for x (the sub / super-level set) is convex. $f(x)$ SUB LEVEL FOR 0

What about these? Feasible region for x is convex in one case, nonconvex in the other.

In general, if f convex, g concave, these are the only relations *guaranteed* to encode a convex feasible region: $f(x) \leq b \quad g(x) \geq b$

Convex Optimisation Forms

$\min_x f(x)$	convex	Assume the domain of all functions is $\mathbb{R}^n$
s.t. $g(x) \leq 0$	convex	State the requirements on the functions in each form in order for the problem to be *guaranteed* convex
$\max_x f(x)$	concave	
s.t. $g(x) \leq 0$	convex	The objective is a convex function (for min) over a convex feasible set.
$\min_x f(x)$	convex	
s.t. $h(x) = 0$, $x \geq 0$	convex and concave: affine! (or if slack variables being used to mimic inequality, can be convex or concave depending)	$Ax - b = 0$
$\min_x f(x)$	convex	
s.t. $g(x) \geq 0$	concave	
$h(x) = 0$	affine	

Rates of Convergence

A more course measure is to look at the limiting behaviour of algorithms:

$\epsilon_k := f(x_k) - f(x^*)$ $\quad$ rate of convergence

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|} = \mu$$

$\mu = 1$	algorithm converges **sublinearly**
$0 < \mu < 1$	algorithm converges **linearly**
$\mu = 0$	algorithm converges **superlinearly**

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^q} < M$$

$M > 0, q > 1$	algorithm converges **superlinearly** with **order q**
$M > 0, q = 2$	algorithm converges **quadratically**

Definiteness of Matrix

$A \in \mathbb{R}^{n\times n}$ $\quad$ an n by n **symmetric** matrix

Positive semidefinite $A \succeq 0$
iff $x^\intercal A x \geq 0 \quad \forall x \in \mathbb{R}^n$

Positive definite $A \succ 0$
iff $x^\intercal A x > 0 \quad \forall x \in \mathbb{R}^n \setminus 0$

Negative (semi-)definite for the opposite direction, and **indefinite** if none of these

$$f(x,y) = x^2 + y^2 + 4xy \qquad [u,v]\begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = 2u^2 + 2v^2 + 8uv$$

Is f convex? ← is $H_f(x,y)$

$$H_f = \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix} \qquad u = 1, v = 1 \implies 2 + 2 + 8 = 12 > 0$$
$$u = -1, v = 1 \implies 2 + 2 - 8 = -4 < 0$$

Constant, so only one Hessian to evaluate

Its Hessian is indefinite, therefore f is non-convex.

Rule-based Step Size for GD

To give you an idea, **two** options from **Barzilai and Borwein 1988** *Two-Point Step Size Gradient Methods:*

1. $\alpha_k := \dfrac{(x_k - x_{k-1})^\intercal(\nabla f(x_k) - \nabla f(x_{k-1}))}{\|\nabla f(x_k) - \nabla f(x_{k-1})\|^2}$ $\quad$ *Smaller steps when gradient is changing rapidly, either in magnitude or direction.*

2. $\alpha_k := \dfrac{\|x_k - x_{k-1}\|^2}{(x_k - x_{k-1})^\intercal(\nabla f(x_k) - \nabla f(x_{k-1}))}$

Use iterate and gradient values from last two iterates.

Newton's Method

For finding roots of f:

$\nabla f(x_k)^\intercal (x_{k+1} - x_k) = -f(x_k)$ $\quad$ if f scalar valued
$J_f(x_k)(x_{k+1} - x_k) = -f(x_k)$ $\quad$ if f vector valued

But we want to find **stationary points not zeros**! **Ideas?**

In order to do this we can replace f with its gradient (assuming it is twice differentiable):

$J_{\nabla f}(x_k)(x_{k+1} - x_k) = -\nabla f(x_k)$ $\quad$ Remember Jacobian of gradient is Hessian.
$\implies H_f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k)$

When adapted to finding zeros of the gradient, it becomes a **second-order** method.

Newton's Method

In order to have a **unique** solution for the Newton step, we need the Hessian to be invertible:

$x_{k+1} - x_k = -H_f(x_k)^{-1}\nabla f(x_k)$

Positive definite matrices are always **invertible**. The implications of this are that a **strictly** convex function will always have a well defined Newton step. If the Hessian is positive **semidefinite**, (function convex but not strictly), then we can have an infinite number of solutions.

Does this make sense? $\quad x^\intercal H_f(x_k) x \geq 0 \quad \forall x \in \mathbb{R}^n$ (pos semidefinite)

This allows our function to have linear segments, where a second derivative doesn't contain any information, so a second-order method doesn't make much sense! For a single-variate function:

$x_{k+1} - x_k = -\dfrac{f'(x_k)}{f''(x_k)}$ $\quad$ Trouble when second derivative is zero (locally linear). Best we can do is a gradient step.

Newton's Method

$$x_{k+1} - x_k = \underbrace{-H_f(x_k)^{-1}\nabla f(x_k)}_{\text{SUGGESTED STEP SIZE / DIRECTION}} \longleftarrow \alpha_k d_k \;(\text{GD})$$

It provides both a direction and a step size. We may want to adjust that step size to improve convergence: we are locally **approximating** the function as being **quadratic** to get our direction and step size but in reality it might not be! Again we can use techniques like **line search**.

What if it **is** quadratic? $\quad f(x) = \frac{1}{2}x^\intercal Q x + c^\intercal x \qquad Q$ pos definite

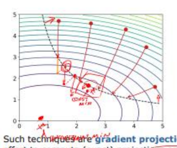
$\nabla f(x) = Qx_k + c \qquad H_f(x_k) = Q$

$H_f^{-1}(x_k)\nabla f(x_k) = Q^{-1}Qx_k + Q^{-1}c = x_k + Q^{-1}c$

$(x_{k+1} - x_k) = -x_k - Q^{-1}c \implies \boxed{x_{k+1} = -Q^{-1}c}$ WHERE YOU WILL MOVE TO

We go straight to the minimum in one step.

Following Gradient to Boundary



Consider various starting points

We would need some way to move along the boundary

Stopping at the intersection of the gradient step with the boundary doesn't work, but if we can take a **projection** of our gradient step back onto our **feasible set** we can make progress.

Such techniques are **gradient projection methods**. It can require extra effort to compute (exact) projections and convergence can be slow.

Lagrangian Relaxation

Just like with duality in LPs, we are going to start off **generating lower bounds** to our optimisation problem. Assume we have a single constraint (g is a scalar valued function):

$\min_x f(x)$ s.t. $g(x) \leq 0$ $\quad \rightarrow \quad \min_x f(x) + \mu g(x)$ $\quad$ for some $\mu \geq 0$

This is a **Lagrangian relaxation** of our constraint.

$\min_x f(x)$ s.t. $g(x) \leq 0$ $\quad \geq \quad \min_x f(x) + \mu g(x)$ $\quad$?

Assuming the problem is feasible with optimal solution: x^*

The relaxation can always pick this point which will be at least as good: $f(x^*) \geq f(x^*) + \mu g(x^*)$ ≤ 0

Because $\quad g(x^*) \leq 0$, $\mu \geq 0$

Lagrangian Dual Problem

$$\min_x f(x) \quad \text{s.t. } g(x) \leq 0 \leftarrow \mu \text{ FOR THIS SET CONS}, \quad h(x) = 0 \leftarrow \lambda$$

Dual variables, one for each constraint again

Also known as **Lagrange** or **KKT multipliers**

Lagrangian Function:

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^\intercal g(x) + \lambda^\intercal h(x) \qquad \mu \geq 0$$

Lagrangian Dual Function:

$$d(\mu, \lambda) = \inf_x \mathcal{L}(x, \mu, \lambda)$$

The dual function is **always** concave!

like a min, see next slide

Dual Problem:

$$\max_{\mu\geq 0, \lambda} d(\mu, \lambda) = \max_{\mu\geq 0, \lambda} \inf_x \mathcal{L}(x, \mu, \lambda)$$

Find tightest lower bound

Slater's Condition (1950)

$$\min_x f(x) \quad \text{s.t. } g(x) \leq 0, \quad h(x) = 0$$

$Ax + b$ $\quad\leftarrow$ generally should be affine for convex problem

For convex optimisation problems it requires that the:

Interior of feasible region is not empty... considering the constraints that are **not** affine / linear:

Assuming all the g constraints are nonlinear, then there exists at least one point x^* where:

$$g(x^*) < 0 \qquad h(x^*) = 0$$

NOT LIN NOR AFFINE — YES — YES BECAUSE ALL CONSTR ARE AFF. — NO DOES NOT SAT. SLATERS — BUT HAS STRONG DUALITY

Dual Function Example

Let's look at an example of this rather than digging into the proof.

$$\min_x (x - 0.6)^2 \;\text{ s.t. } 0 \leq x \leq 1,\; x \in \mathbb{Z} \quad = \quad \min_x (x-0.6)^2 \;\text{ s.t. } 0 \leq x \leq 1,\; x(x-1) = 0,\; x \in \mathbb{R}$$

Reformulate integrality requirement as a nonconvex constraint

$$\max_{\lambda\in\mathbb{R}} \min_{x\in\mathbb{R}} (x-0.6)^2 + \lambda x(x-1) \quad \text{s.t. } 0 \leq x \leq 1$$

"Dual problem" (to keep simple we only relax the nonconvex constraint)

KKT Conditions

PRIMAL POINT

For x^* to be a local minimum, there exist some KKT multipliers $\mu\ \lambda$ where:

$g(x) \leq 0 \; (\mu)$, $h(x) = 0 \; (\lambda)$

$\nabla_x \mathcal{L}(x^*, \mu, \lambda) = 0$	(stationarity) — UNCONSTR: $\nabla_x \lambda = 0$
$g(x^*) \leq 0 \qquad h(x^*) = 0$	(primal feasibility) — TRIVIAL: x^* MUST BE A FEAS POINT
$\mu \geq 0$	(dual feasibility) — LAG DUAL PROB
$\mu_j g_j(x^*) = 0$	(complementary slackness) — IF g_j IS NOT ACTIVE $g_j(x) < 0$ THEN μ_j MUST BE 0

$$\nabla f(x^*) + J_g(x^*)^\intercal \mu + J_h(x^*)^\intercal \lambda = 0$$

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{i=1}^{p} \lambda_i \nabla h_i(x^*) = 0$$

KKT Interpretation

We can consider the dual variables as the force at which a constraint pushes back preventing us from getting a better solution.

Stationarity: A stationary point in x for the Lagrangian. The "force" due to the objective is exactly balanced by constraints preventing a better solution.

$$\underbrace{\nabla f(x^*)}_{\text{FORCE TO IMPROVE OBJ}} + \underbrace{\sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{i=1}^{p} \lambda_i \nabla h_i(x^*)}_{\text{FORCE TO REMAIN FEAS}} = 0$$

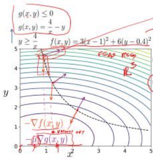
Primal feasibility: We are in the feasible region.

Dual feasibility: An inequality can only block in one direction. $\mu \geq 0$

Complementary Slackness: Constraint can only push back if we are touching it. $\quad \mu_j g_j(x^*) = 0$

KKT Stationarity

$g(x,y) \leq 0$, $\quad g(x,y) = \frac{4}{x} - y$, $\quad y \geq \frac{4}{x}$, $\quad f(x,y) = 3(x-1)^2 + 6(y-0.4)^2$



$\nabla_{x,y}\mathcal{L}(x,y,\mu) = \nabla f(x,y) + \mu \nabla g(x,y)$

$$\nabla f(x,y) = \begin{bmatrix} 6(x-1) \\ 12(y-0.4) \end{bmatrix} \qquad \nabla g(x,y) = \begin{bmatrix} -\frac{4}{x^2} \\ -1 \end{bmatrix}$$

(x,y) $\qquad \nabla f(x,y) + \mu \nabla g(x,y) = 0$ $\quad$ DOES NOT SAT. STAT. COND

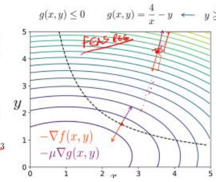
$(1,4) \rightarrow \begin{cases} 0 - 4\mu = 0 \implies \mu = 0 \\ 43.2 - \mu = 0 \implies \mu = 43.2 \end{cases}$

$(1.5, 2.67) \rightarrow \begin{cases} 3 - 1.78\mu = 0 \implies \mu = 1.69 \\ 27.2 - \mu = 0 \implies \mu = 27.2 \end{cases}$

$(2.512, 1.592) \rightarrow \begin{cases} 9.07 - 0.634\mu = 0 \implies \mu = 14.3 \\ 14.3 - \mu = 0 \implies \mu = 14.3 \end{cases}$

KKT Complementary Slackness

$$g(x,y) \leq 0 \qquad g(x,y) = \frac{4}{x} - y \longleftarrow y \geq \frac{4}{x} \qquad f(x,y) = 3(x-1)^2 + 6(y-0.4)^2$$



$$\nabla f(x,y) = \begin{bmatrix} 6(x-1) \\ 12(y-0.4) \end{bmatrix} \qquad \nabla g(x,y) = \begin{bmatrix} -\frac{4}{x^2} \\ -1 \end{bmatrix}$$

Complementary slackness:

$$\mu g(x,y) = 0$$
$$\mu\left(\frac{4}{x} - y\right) = 0$$

Prevents solutions such as...

Interior Point Method

Barrier functions introduced to objective that approach infinity at inequality constraint boundary.

Progressively move towards a solution that satisfies the KKT conditions, rather than attempting to directly solve them.

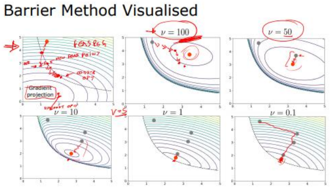
We will outline the algorithm (with some simplifications) that lies behind **Ipopt**. This is a **primal-dual** interior point method. More details:

FOSS

***Wachter and Biegler 2006** On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming*

Generate unconstraint problem and over time we ensure that the problem remains feasible

Barrier Method Visualised



Primal-Dual IPM / BM

We can gradually solve the **primal** and **dual** variables for the problem, warmstarting the barrier problem with both for each update of ν

We are going to solve problem in the following form:

$$\min_x f(x) \;\text{ s.t. } h(x) = 0,\; x \geq 0 \quad \xrightarrow{\text{Barrier functions applied to just inequality constraints:}} \quad \min_x \phi(x, \nu) := f(x) - \nu \sum_j^n \ln(x_j) \;\text{ s.t. } h(x) = 0$$

Complementary Slackness $(\mu_j g_j(x^*) = 0)$ is not need for problems in this form

The barrier problem is an equality **constrained problem**, that we solve by applying **Newton's Method** to its **KKT conditions**.

Solving the Barrier Problem

$$s(x,\lambda) := \nabla f(x) - \nu x^{-1} + \sum_i^m \lambda_i \nabla h_i(x) = 0 \quad \leftarrow \text{n equality constraints one for each primal variable}$$
$$h(x) = 0 \quad \leftarrow \text{m equality constraints}$$

We employ Newton's method to iteratively solve this system of nonlinear equations.

$$J_u(x_k, \lambda_k)\begin{bmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \end{bmatrix} = -u(x_k, \lambda_k) \qquad u(x_k, \lambda_k) := \begin{bmatrix} s(x_k, \lambda_k) \\ h(x_k) \end{bmatrix}$$

$(n+m) \times (n+m)$

$$J_u(x_k, \lambda_k) = \begin{bmatrix} J_{s,x}(x_k, \lambda_k) & J_{s,\lambda}(x_k, \lambda_k) \\ J_{h,x}(x_k) & J_{h,\lambda}(x_k) \end{bmatrix} = \begin{bmatrix} W_k & A_k^\intercal \\ A_k & 0 \end{bmatrix}$$

JACOBIAN OF GRAD IS THE HESSIAN

$$W_k := H_{\mathcal{L},x}(x_k, \lambda_k) \qquad A_k := J_{h,x}(x_k)$$

Link to Original KKT

$$s(x,\lambda) := \nabla f(x) - \nu x^{-1} + \sum_i^m \lambda_i \nabla h_i(x) = 0$$
$$\nabla f(x) - \mu + \sum_i^m \lambda_i \nabla h_i(x) = 0 \qquad \text{Gradient of Lagrangian} \qquad \mu = \nu x^{-1} \quad \mu_j x_j = \nu$$

DUAL VAR FOR $x \geq 0$

We can rewrite the KKT conditions for the barrier problem as:

$$\nabla f(x) - \mu + \sum_i^m \lambda_i \nabla h_i(x) = 0$$
$$h(x) = 0$$
$$\mu_j x_j = \nu$$

WITH COMPL. SLACKNESS FOR $x \geq 0$ RELAXED

As ν approaches zero, and keeping $x \geq 0$ $\;\mu \geq 0$, these become the KKT conditions for the original problem! $\quad g(x) := -x \leq 0$

The barrier method gradually enforces the nasty complementary slackness term.