Structured Problems and Algorithms

Integer and quadratic optimization problems

Jim Thrasher

Dept. of Engg. and Comp. Sci.,
Univ. of Cal., Davis

Aug. 13, 2010

Table of contents

Benefits of Structured Problems

Optimization problems may become more tractable if we take advantage of the *structured* aspects of the problem rather than using *generalized* approaches. Methods for solving a variety of structured problems are described in this survey.

1. *Integer Programming*: Some or all design variables must be integer-valued. Both pure and mixed integer categories.
2. *Quadratic Programming*: Quadratic objective functions and linear constraints.

Integer programming overview

Why integer-valued optimization?

1. Many practical problems require integer-valued solutions.
2. Can't use LP solution when problem is constrained by small integer bounds.
3. Binary variables useful in selecting optimal mix of several candidate product inputs, task selection, or investment candidates.

Integer-constrained optimization

Integer programming problems are combinatorial as opposed to continuous so solutions can involve exhaustive search of finite but very large candidate sets. A very important precondition for solving integer programs is reducing the size of the search set.

1. Solve LP, then truncate or round fractional values of LP optimum.
2. Maintain feasibility.
3. Enforce integer constraints only if $x_i < 20$.
4. Search reduced set of candidate solutions.
5. Select one that is feasible and closest to LP optimum.

In practice we often accept as optimal integer solutions whose objective function is within a few percent of the optimal solution. This may make the difference between an *acceptable* solution or *NO* solution!

Structured Problems and Algorithms

0-1 Integer Problems

Example 11.1

Fixed-Charge Problem. Given a fixed number of products, made from a fixed number of restricted-availability resources, determine the optimal profitability of production.

	N products
	K_j fixed setup cost for j^{th} product
	C_j variable cost per unit of j^{th} product
	M resources, a_{ij} units of resource i for prod. j.
demand	Demand for prod. j is d_j, sells for p_j per unit.
res. limits	Max. of b_i units of resource i available $(i = 1, 2, \ldots, M)$

Example 11.1

Formulation

δ_j Cost of production is nonlinear. Linearize w/binary variables.

Z Objective function: aggregate net profit.

$\delta_j = 0$ Not producing some of the N products may result in optimum obj. fcn.

decision $\delta_j = \left\{ \begin{matrix} 1 & \text{if product j is produced} \\ 0 & \text{otherwise} \end{matrix} \right\}$

profit Maximize $Z = \sum_{j=1}^{N} p_j x_j - \sum_{j=1}^{N} (K_j \delta_j + c_j x_j)$

supply $\sum_{j=1}^{N} a_{ij} x_j \leq b_j \quad (i = 1, 2, \ldots, M)$

demand$_1$ $x_j \leq d_j \delta_j \quad (j = 1, 2, \ldots, N)$

demand$_2$ $x_j \geq 0$ and $\delta_j \in \{0, 1\} \quad \forall j \in \{j = 1, 2, \ldots, N\}$

Example 11.2

A nonlinear 0-1 (binary) integer programming problem.

$$\begin{aligned} \text{Maximize} \quad & Z = x_1^2 + x_2 x_3 - x_3^3 \\ \text{Subject to} \quad & -2x_1 + 3x_2 + x_3 \leq 3 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{aligned}$$

Note that for any binary variable, x_j, $x_j^k = x_j$. The objective function reduces to:

$$Z = x_1 + x_2 x_3 - x_3$$

$x_2 x_3$ remains nonlinear. Introduce $y_1 = x_2 x_3$ as a new binary variable. Want $y_1 = 1$ *only* when $x_2 = 1 \wedge x_3 = 1$. Use following constraints:

$$\begin{aligned} x_2 + x_3 - y_1 &\leq 1 & \quad x_2 = x_3 = 1 \quad & y_1 \geq 1 \wedge y_1 \leq 1 \Rightarrow y_1 = 1 \\ -x_2 - x_3 + 2y_1 &\leq 0 & \quad x_2 = 0 \vee x_3 = 0 \quad & y_1 \leq \frac{x_2 + x_3}{2} \Rightarrow y_1 = 0 \end{aligned}$$

Example 11.2

$$\text{(Original)} \quad Z = x_1^2 + x_2 x_3 - x_3^3$$

Fully linearized 0-1 problem.

$$\begin{aligned} \text{Maximize} \quad & Z = x_1 + y_1 - x_3 \\ \text{Subject to} \quad & -2x_1 + 3x_2 + x_3 \leq 3, \\ & x_2 + x_3 - y_1 \leq 1, \\ & -x_2 - x_3 + 2y_1 \leq 0, \\ & x_1, x_2, x_3, y_1 \in \{0, 1\} \end{aligned}$$

1. Use continuous variables rather than integer.
2. Generalize method for transforming products of binary vars.

Structured Problems and Algorithms

Branch-and-Bound Algorithm

use Most widely used for solving pure and mixed IPs.

use Most commercial packages are based on B-and-B.

approach A method for sharply reducing solution search set.

approach LPs with continuous variables are methodically transformed into MIPs.

Branch-and-Bound Algorithm

- If LP optimal solution contains fractional values for some integer vars., then these variables must be integers for the optimal integer solution.
- A reasonable search will begin with integers near to the LP result, e.g. floor or ceiling for the variable under consideration.
- But it may be that the optimal integer solution for a given variable will not even be the nearest integer greater or less than the LP result.
- Branch-and-bound systematically explores integer-valued candidates near the LP optimum values.
- Even though B-and-B eliminates a very large percentage of 'possible' integer-valued inputs, the total computation can remain very expensive.

Branch-and-Bound Algorithm

Example Mixed-integer program

Solve the following MIP using the branch-and-bound method:

Maximize $Z = 3x_1 + 2x_2$

C_1 $x_1 \leq 2$

C_2 $x_2 \leq 2$

C_3 $x_1 + x_2 \leq 3.5$

C_4 $x_1, x_2 \in \mathbb{N}$

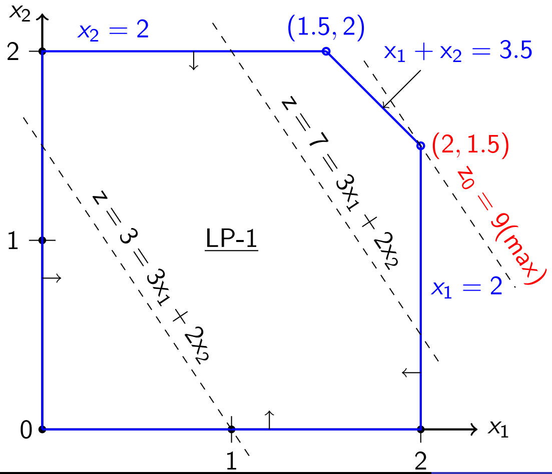
First, solve this as an LP, allowing:

C_4^{LP} $x_1, x_2 \in \mathbb{R}$

The LP optimal solution is: $x_1 = 2, x_2 = 1.5$.
The maximum of the objective function, Z, is: $Z_0 = 9$.
The following figure illustrates this result.

Branch-and-Bound Algorithm

LP-1



Maximize $Z = 3x_1 + 2x_2$

C_1 $x_1 \leq 2$

C_2 $x_2 \leq 2$

C_3 $x_1 + x_2 \leq 3.5$

Solution:

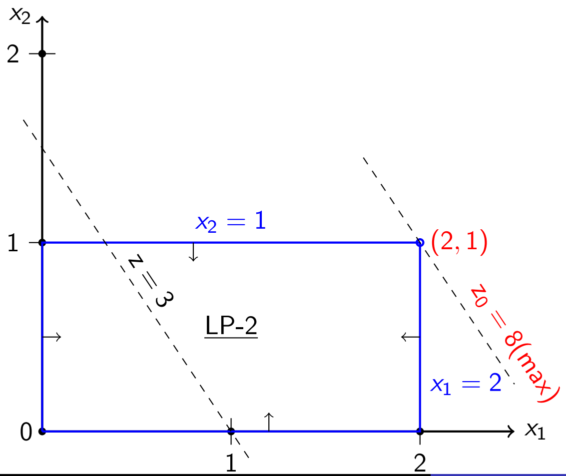
$x_1 = 2$

$x_2 = 1.5$

Obj $Z_0 = 9$

- *NOT* optimal for MIP.
- $Z_0 = 9$ is an *upper bound.*
- Now try int. vals. for x_2

Branch-and-Bound Algorithm
LP-2



Maximize $Z = 3x_1 + 2x_2$

C_1 $x_1 \leq 2$

C_2 $x_2 \leq 1$

C_3 $x_1 + x_2 \leq 3.5$

C_4 $x_1, x_2 \in \mathbb{N}$

Solution:

$x_1 = 2$

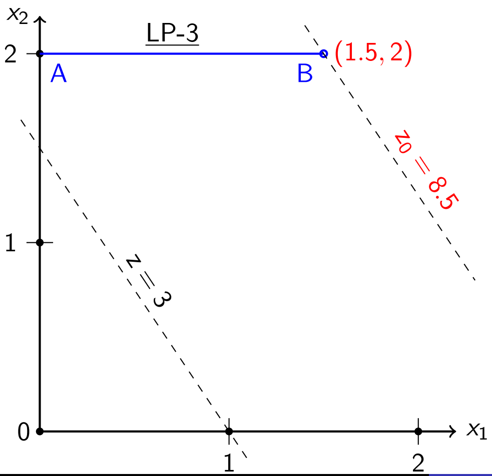
$x_2 = 1$

Obj $Z_0 = 8$

$Z_0 = 8$ is a *lower bound*.
Now try $x_2 \geq 2$

Branch-and-Bound Algorithm

LP-3



Maximize $Z = 3x_1 + 2x_2$

C_1 $x_1 \leq 2$

C_2, C_2a $x_2 \leq 2 \wedge x_2 \geq 2$

C_3 $x_1 + x_2 \leq 3.5$

C_4 $x_2 \in \mathbb{N}$

Solution:

$x_1 = 1.5$

$x_2 = 2$

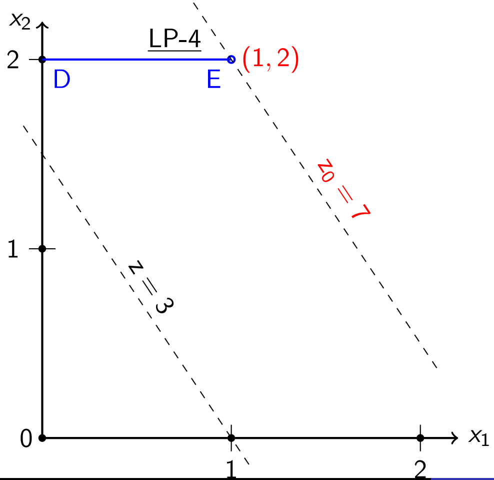
Obj $Z_0 = 8.5$

$Z_0 = 8.5$ not *feasible*

$x_1 = 1.5 \notin \mathbb{N}$

Branch-and-Bound Algorithm

LP-4



Maximize $Z = 3x_1 + 2x_2$

C$_1$ $x_1 \leq 1$

C$_2$ $x_2 \leq 2 \wedge x_2 \geq 2$

C$_3$ $x_1 + x_2 \leq 3.5$

C$_4$ $x_1, x_2 \in \mathbb{N}$

Solution:

$x_1 = 1$

$x_2 = 2$

Obj $Z_0 = 7$

$Z_0 = 7 <$ *lower bound*

$x_1 \geq 2$ violates C$_3$.

Structured Problems and Algorithms

Structure of branch and bound solution

Tree-structured solution process

The process of solving a MIP by branch and bound method results in a tree structure whose nodes represent tentative choices for the set of integer constraints to the original LP problem at the root node.

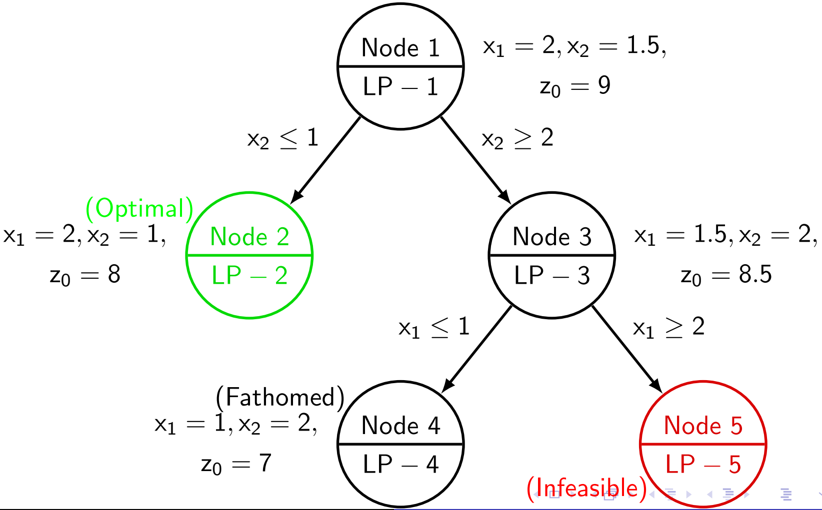
Tree Diagram of branch and bound procedure

- Node 1: initial LP problem (LP-1) $\rightarrow$ Frac
- Node 2: constrain $x_2 \leq 1$ (LP-2) $\rightarrow$ Int
- Node 3: constrain $x_2 \geq 2$ (LP-3) $\rightarrow$ Frac
- Node 4: constrain $x_1 \geq 2$ (LP-4) $\rightarrow$ Int
- Node 5: constrain $x_1 \geq 2$ (LP-5) $\rightarrow$ IF

After all nodes are *fathomed*, we select the best of the *integer* solutions: (LP-2). The next slide illustrates this result...

Branch-and-Bound Algorithm: LP-1...5

Tree diagram



Steps of branch and bound method (MIP)

Initial process

Initial solution bounding...

- Solve MIP as LP-1 w/o integer restrictions, yielding Z_1.
- Take Z_1 as an *upper bound* on the optimum *integer* solution.
- Partition feasible region of (LP-1) by branching on an integer variable at a fractional value.
- Seek a feasible integer solution to serve as an initial *lower bound*

Partitioning rules...

- Select IV with largest fractional value in the LP solution
- Assign priorities on IVs. Branch on top priority one.
  - (a) Represents an important decision in model
  - (b) Cost/profit coeff. is very large
  - (c) Value high based on past experience
- Branch on arbitrary choice.

Next we illustrate how this might proceed...

Steps of branch and bound method

Fathoming and solution rules

Selecting a branch node...

1. Use Optimal Value of Obj F Branch from node w/max Z.
2. Last-In-First-Out Rule Branch from most recently solved LP.

Finding first integer solution...(this might be the hardest step)

DO Choose IV, x_j w/fractional value

Solve LP In feasible region w/x_j integer constrained

Add node Check if integer solution obtained yet

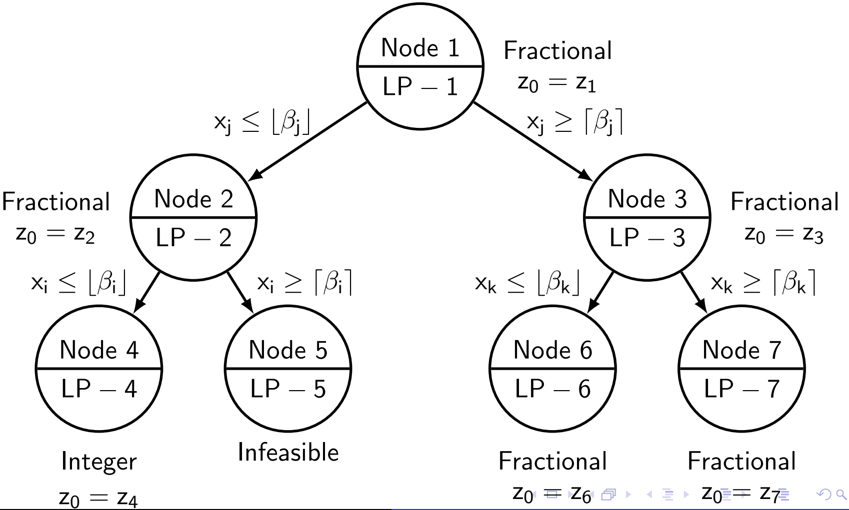
WHILE No integral solution for Z found.

Set lwr bound Z_{lb} is the highest-valued int. sol.

Continue fathoming nodes that might yield $Z > Z_{lb}$, until all paths blocked. This is illustrated in the next frame (Fig. 11.6)...

Branch-and-Bound Algorithm: LP-1...7

Tree diagram (general)



Steps of branch and bound method

Fathoming and solution rules

Solution Time

The time it takes to solve the Integer Problem is *very* sensitive to how the problem is formulated initially!

Guidelines based on practical experience at IBM Research:

1. Keep set of IV's as small as possible: No int constraints if $x_i > 20$.
2. Provide tight lwr/upr bound on integer variables when possible.
3. Freely add new (integer) constraints to *reduce* total solution time.
4. Accept first int solution that is within 0.03 of continuous optimum.
5. Choose branching variables based on experience-dictated importance.

Branch-and-bound can also be used to solve nonlinear IP. See Gupta and Ravindran for implementation of B-and-B using generalized reduced gradient (GRG). They also provide a good survey (ca. 1981) of B-and-B heuristics.

Branch and bound for nonlinear programs

General method [*Gupta and Ravindran, 1985*]

The general branch and bound method also handles convex *nonlinear* mixed integer programs (NLMIP).

Nonlinear mixed integer program definition

$$
\begin{array}{lll}
\text{Maximize:} & f(x), & x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n, \\
\text{Subject to:} & g_i(x) \geq 0, & i = 1, 2, \ldots, NI, \\
 & h_k(x) = 0, & k = 1, 2, \ldots, NE, \\
 & x_j \in \mathbb{Z} & j = 1, 2, \ldots, m \leq n
\end{array}
$$

- The first m of $x_1, x_2, \ldots, x_n$ are identified as IV.
- Selecting which values to restrict to integers depends on the problem. The nonlinearity may increase difficulty.
- Solving NLMIP ignoring integer restrictions satisfies first order sufficient optimality conditions.
- Hence a Kuhn-Tucker point is a global extremum.

Structured Problems and Algorithms

What is a quadratic program?

QP definition

A QP problem is an optimization problem involving a *quadratic* objective function and *linear* constraints.

QP problem (scalar)

$$\text{Minimize:} \quad f(x) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} q_{ij} x_{ij}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \text{for } i = 1, \ldots, m$$

$$x_j \geq 0$$

QP problem (vector)

$$\text{Minimize:} \quad f(x) = cx + x^{\mathrm{T}}\mathbf{Q}x \qquad c_{(1\times n)},\ x_{(n\times 1)},\ \mathbf{Q}_{(n\times n)}$$

$$\text{Subject to:} \quad \mathbf{A}x = b \qquad b_{(m\times 1)},\ \mathbf{A}_{(m\times n)}$$

$$x \geq 0$$

Structured Problems and Algorithms

Applications of QP 1

Portfolio Selection with risk minimization

Selecting a portfolio of securities to generate a 'maximum return' while 'minimizing risk' results in a Quadratic Program formulation.

QP for risk-averse portfolio

Minimize:	$Z = x^{\mathrm{T}}\mathbf{Q}x$	Minimize overall risk.
Subject to:	$\sum_{j=1}^{N} x_j \leq C$	A minimum requred cash reserve.
	$x_j \geq 0$	Only positive amounts invested in x_j
	$\mu^{\mathrm{T}}x \geq R$	Total return must be at least \$ R

- $\mathbf{Q}_{(N\times N)} = [q_{ij}]$ is the variance-covariance matrix of the N securities.
- μ^{T} is the average annual return for the securities.

Applications of QP 2

Constrained regression problem

The basic regression problem requires minimizing an error vector that arises from the stochastic relation between independent and dependent variables of the problem.

L-S regression problem

Minimize:	$f(x) = e^{\mathrm{T}}\mathbf{I}e$	$e_{(n\times 1)}$
Subject to:	$Y = \mathbf{X}\beta + e$	$\mathbf{X}_{(n\times m)}$, $\beta_{(m\times 1)}$

Types of additional constraints applied:

- $\beta_j \geq 0$ and $\sum \beta_j = 1$.
- Error terms weighted: $e^{\mathrm{T}}\mathbf{A}e$ where $\mathbf{A}$ is the weighting matrix

Structured Problems and Algorithms

Kuhn-Tucker Conditions

Quadratic programs are a restriction of general nonlinear programs. The QP optimization is of the form:

KTC for QP

Minimize: $f(x) = cx + x^{\mathrm{T}}\mathbf{Q}x$

Subject to: $g(x) = x \geq 0$ Inequality constraints.

$h(x) = \mathbf{A}x - b = 0$ Equality constraints.

The associated Kuhn-Tucker conditions for QPs:

Solve: $c + x^{\mathrm{T}}(\mathbf{Q} + \mathbf{Q}^{\mathrm{T}}) - u - v\mathbf{A} = 0$

$\mathbf{A}x = b$ $\quad x \geq 0$

$ux = 0$ $\quad u \geq 0$

$u \geq 0$ $\quad v$ unrestricted in sign

QP: Kuhn-Tucker Conditions

KTC Example 11.6

Illustration of Kuhn-Tucker conditions for a Quadratic Program.

Minimize: $f(x) = -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2$

Subject to: $x_1 + x_2 = 2 \qquad x_1, x_2 \geq 0$

The coefficient arrays and vectors are:

$$c = (-6, 0) \quad \mathbf{Q} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad \mathbf{A} = (1, 1) \quad b = (2)$$

The Kuhn-Tucker conditions are

$$(-6, 0) + (x_1, x_2) \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix} - (u_1, u_2) - v_1(1, 1) = 0$$

$$-6 + 4x_1 - 2x_2 - u_1 - v_1 = 0$$

$$0 - 2x_1 + 4x_2 - u_2 - v_1 = 0$$

$$x_1 + x_2 = 2 \qquad x_1, x_2 \geq 0$$

$$u_1x_1 = 0 \qquad u_2x_2 = 0$$

$$u_1, u_2 \geq 0 \quad v_1 \text{ unrestricted in sign}$$

QP: Kuhn-Tucker Conditions

KTC Example 11.6

What this example illustrates...

1. Because the constraints of QP are linear, the constraint qualification is *always* satisfied $\rightarrow$ Kuhn-Tucker necessity theorem applies.
2. Therefore the Kuhn-Tucker conditions are *necessary* for the optimality of QP.
3. When $\mathbf{Q}$ is positive (semi-)definite the objective function is convex and KTC are *sufficient* for optimum solution of QP.
4. Then solving KTC finds an optimal solution to QP.

In 1959, Wolfe first used phase I simplex method to solve KTC for QP. It was only effective in solving QP for $\mathbf{Q}$ positive definite. It fails to converge for $\mathbf{Q}$ positive semidefinite. Lemke (1965) introduced the *complementary pivot method* which allows us to solve QP efficiently when $\mathbf{Q}$ is positive semidefinite.

Structured Problems and Algorithms

Complementary Pivot Method

Complementary problem

The *complementary pivot method* for solving QPs uses the notion of *complementary problem*.

Complementary problem

Solve: $w = \mathbf{M}z + q \qquad (w, z, q)_{(n\times 1)},\ \mathbf{M}_{(n\times n)}$

Subject to: $w \geq 0,\ z \geq 0$

$w'z = 0$

Note: no objective function to optimize. Also...

1. Solving a system of simultaneous linear equations.
2. A valid solution will be nonnegative.
3. A single nonlinear constraint.

Complementary Pivot Method

Complementary problem - sample

Consider a problem with

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{pmatrix} \qquad q = \begin{pmatrix} 2 \\ -5 \\ -3 \end{pmatrix}$$

The *complementary problem*: Find $w_1, w_2, w_3, z_1, z_2, z_3$ s.t.

$$\begin{aligned} w_1 &= z_1 + 2z_2 + 3z_3 + \ 2 \\ w_2 &= 4z_1 + 5z_2 + 6z_3 - 5 \\ w_3 &= 6z_1 + 7z_2 + 8z_3 - 3 \end{aligned}$$

constrained by:

$$w_1, w_2, w_3, z_1, z_2, z_3 \geq 0$$

$$w_1 z_1 + w_2 z_2 + w_3 z_3 = 0$$

The optimal solution to a QP may be obtained from an equivalent complementary problem formed from the KTC of the QP.

Complementary Pivot Method

Consider a convex quadratic problem of the form

Convex QP

Minimize: $f(x) = cx + x^{\mathrm{T}}\mathbf{Q}x \quad \mathbf{Q}_{(n\times n)}$, symm, pos def/semidef

Subject to: $\mathbf{A}x \geq b$ and $x \geq 0$

The KTC optimality conditions to the above system can be written

KTC of Convex QP

Solve: $u = 2\mathbf{Q}x - \mathbf{A}^{\mathrm{T}}y + c^{\mathrm{T}}$

$v = \mathbf{A}x - b$

Subject to: $x, y, u, v \geq b$ and $u^{\mathrm{T}}x + v^{\mathrm{T}}y = 0$

To cast it as a complementary problem, we use

$$w = \begin{pmatrix} u \\ v \end{pmatrix} \quad z = \begin{pmatrix} x \\ y \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 2\mathbf{Q} & -\mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \quad q = \begin{pmatrix} c^{\mathrm{T}} \\ -b \end{pmatrix}$$

Complementary Pivot Method

Convex QP Example 11.8

Illustration of Kuhn-Tucker conditions for a Quadratic Program.

Minimize: $f(x) = -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2$

Subject to: $-x_1 - x_2 \geq -2 \quad \text{and} \quad x_1, x_2 \geq 0$

The coefficient arrays and vectors are:

$$\mathbf{A} = (-1, -1) \quad b = (-2) \quad c^{\mathrm{T}} = \begin{pmatrix} -6 \\ 0 \end{pmatrix} \quad \mathbf{Q} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

The equivalent complementary problem is given by

$$\mathbf{M}_{(3\times3)} = \begin{pmatrix} \mathbf{Q} + \mathbf{Q}^{\mathrm{T}} & -\mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} 4 & -2 & 1 \\ -2 & 4 & 1 \\ -1 & -1 & 0 \end{pmatrix}, \text{ and } q = \begin{pmatrix} c^{\mathrm{T}} \\ -b \end{pmatrix} = \begin{pmatrix} -6 \\ 0 \\ 2 \end{pmatrix}$$

The values of z_1 and z_2 correspond to the optimal values of x_1 and x_2 since the QP matrix $\mathbf{Q}$ is positive definite.

Complementary Pivot Method

Development of method - 1

Complementary problem

Solve: $w = \mathbf{M}z + q \qquad (w, z, q)_{(n\times 1)},\ \mathbf{M}_{(n\times n)}$

Subject to: $w \geq 0,\ z \geq 0$

$w^{\mathrm{T}}z = 0$

1. A nonnegative solution (w, z) to the CP is called a *feasible solution*
2. A feasible solution (w, z) to the CP that also satisfies the complementarity condition $w^{\mathrm{T}}z = 0$ is called a *complementary solution*.

Remarks...

- $w^{\mathrm{T}}z = 0 \equiv \forall i : w_i z_i = 0$.
- (w_i, z_i) is called a *complementary pair*.
- If $q \geq 0$, one solution is $w = q, z = 0$.
- Want at least one element of q negative for nontrivial solution.
- In that case, $w = q, z = 0$ is *infeasible* to CP.

Complementary Pivot Method
Development of method - 2

The complementary pivot method posed by Lemke (1965)

- Begin with the infeasible solution $w = q, z = 0$.
- Introduce the artificial variable z_0.
- This will result in an *almost complimentary solution*.

Basic solution

$$z_0 = -\min(q_i) \quad \text{in}$$

$$w_i = q_i + z_0 \qquad z_i = 0 \quad \forall i = 1, \ldots, n$$

This is a solution, but it is not feasible.
We call it an *almost complimentary solution*

The original equation system must be augmented before we can apply the complementary pivot method...

Complementary Pivot Method

Development of method - 3

Augmented system

Solve: $w - \mathbf{M}z - ez_0 = q$

Subject to: $w, z, z_0 \geq 0$

$w^{\mathrm{T}}z = 0$

where $e_{(n\times 1)} = (1, 1, \ldots, 1)^{\mathrm{T}}$

The initial tableau becomes (q_s is the most negative element of q):

Initial Tableau

Basis	w_1	$\cdot$	w_s	$\cdot$	w_n	z_1	$\cdot$	z_s	$\cdot$	z_n	z_0	q
w_1	1					$-m_{11}$		$-m_{1s}$		$-m_{1n}$	-1	q_1
w_s			1			$-m_{s1}$		$-m_{ss}$		$-m_{sn}$	-1	q_s
w_n					1	$-m_{n1}$		$-m_{ns}$		$-m_{nn}$	-1	q_n

Complementary Pivot Method

Algorithm - 1

z_0 replaces w_s from the basis. Performing the pivot operation yields the new tableau where

$$q'_s = -q_s \qquad q'_i = q_i - q_s, \qquad \forall i \neq s$$

$$m'_{sj} = \frac{-m_{sj}}{-1} = m_{sj} \qquad \forall j = 1, \ldots, n$$

$$m'_{ij} = -m_{ij} + m_{sj} \qquad \forall j = 1, \ldots, n \quad \text{and} \quad i \neq s$$

Tableau after first pivot

Basis	w_1	$\cdot$	w_s	$\cdot$	w_n	z_1	$\cdot$	z_s	$\cdot$	z_n	z_0	q
w_1	1		-1		0	$-m'_{11}$		$-m'_{1s}$		$-m'_{1n}$	0	q'_1
z_0	0		-1		0	$-m'_{s1}$		$-m'_{ss}$		$-m'_{sn}$	1	q'_s
w_n	0		-1		1	$-m'_{n1}$		$-m'_{ns}$		$-m'_{nn}$	0	q'_n

Complementary Pivot Method
Algorithm - 2

The complementary pivot method in a finite number of pivot transformations leads from an *almost complementary solution* to a *complementary solution*.

Notes on pivot series

1. $q_i' \geq 0, i = 1, \ldots, n.$
2. Basic solution, $w_1 = q_1', \ldots, w_{s-1} = q_{s-1}', z_0 = q_s', \ldots, wn = q_n'$, and all other vars initially 0 is an almost complementary solution.
3. The almost complementary solution becomes a complementary solution as soon as z_0 is reduced to 0.
4. In each step, maintain $w_i z_i = 0$ for all $i = 1, \ldots, n$.
5. Basic solution remains ≥ 0: q_i's must be ≥ 0 in all tableaus.

Complementary Pivot Method

Algorithm - 3

Step 2 and 3

1. *complementary rule*: Add complement to basis of variable that just left basis in last tableau, e.g. if w_s was taken out, bring z_s in.
2. Before adding new basic variable, determine which to remove using *minimum-ratio* test. Find $\min_{m_{is}>0} \frac{q'_i}{m'_{is}}$.
3. The index from the minimum ratio test, k, determines which w_k to replace by the complementary z_k.
4. (Step 3) Now that z_k has entered basis, continue pivot and replacement process until the termination condition.

Termination conditions

1. Minimum ratio obtained in row s and z_0 leaves basis. After pivot, we have the complementary solution
2. Minimum ratio fails $\rightarrow$ no solution to complementary problem exists.

Complementary Pivot Method

Summary

Remarks

1. The complementary pivot method always terminates with a complementary solution in a finite number of steps whenever
   - (a) all elements of $\mathbf{M}$ are positive, or
   - (b) $\mathbf{M}$ has positive principal determinants (incl. $\mathbf{M}$ positive definite).
2. The minimum-ratio test fails, since all coeff. in the pivot column are nonpositive. This implies no solution: CP has a *ray solution*

See Cottle and Dantzig (1967) for a full development with proofs.

Structured Problems and Algorithms

Complementary Pivot Method

Example of algorithm

Tableau 2

Basis	w_1	w_2	w_3	z_1	z_2	z_3	z_0	q
w_1	1	0	0	-4	2	-1	-1	-6
w_2	0	1	0	2	-4	-1	-1	0
w_3	0	0	1	1	1	0	-1	2

Tableau 3

Basis	w_1	w_2	w_3	z_1	z_2	z_3	z_0	q
z_0	-1	0	0	4	-2	1	1	6
w_2	-1	1	0	6	-6	0	0	6
w_3	-1	0	1	6	-1	1	0	8

Complementary Pivot Method

Example of algorithm

Tableau 4

Basis	w_1	w_2	w_3	z_1	z_2	z_3	z_0	q
z_0	$-\frac{1}{3}$	$-\frac{2}{3}$	0	0	2	1	1	2
z_1	$-\frac{1}{6}$	$\frac{1}{6}$	0	1	-1	0	0	1
w_3	$-\frac{1}{6}$	$-\frac{5}{6}$	1	0	4	1	0	3

Tableau 5

Basis	w_1	w_2	w_3	z_1	z_2	z_3	z_0	q
z_0	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	1	$\frac{1}{2}$
z_1	$-\frac{5}{24}$	$-\frac{1}{24}$	$\frac{1}{4}$	1	0	$\frac{1}{4}$	0	$\frac{7}{4}$
z_2	$-\frac{5}{24}$	$-\frac{5}{24}$	$\frac{1}{4}$	0	1	$\frac{1}{4}$	0	$\frac{3}{4}$

Complementary Pivot Method

Example of algorithm

Tableau 6

Basis	w_1	w_2	w_3	z_1	z_2	z_3	z_0	q
z_3	$-\frac{1}{2}$	$-\frac{1}{2}$	-1	0	0	1	2	1
z_1	$-\frac{1}{12}$	$-\frac{1}{12}$	$\frac{1}{2}$	1	0	0	$-\frac{1}{2}$	$\frac{3}{2}$
z_2	$-\frac{1}{12}$	$-\frac{1}{12}$	$\frac{1}{2}$	0	1	0	$-\frac{1}{2}$	$\frac{1}{2}$