ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

LÊ NHỰT NAM

XẤP XỈ ĐẠO HÀM BẰNG ĐẠO HÀM CỦA ĐA THỨC NỘI SUY

TIỂU LUẬN GIẢI TÍCH SỐ

TP. Hồ Chí Minh - Năm 2024

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

LÊ NHỰT NAM

XẤP XỈ ĐẠO HÀM BẰNG ĐẠO HÀM CỦA ĐA THỨC NỘI SUY

TIỂU LUẬN GIẢI TÍCH SỐ

Giáo viên hướng dẫn: **TS. TRỊNH ANH NGỌC**

TP. Hồ Chí Minh - Năm 2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến thầy *Nguyễn Lê Hoàng Anh* đã cho phép và tạo mọi điều kiện cho chúng em tham gia học dự thính môn học Thuật toán tối ưu trong học phần Cao học vừa qua. Trong quá trình tham gia học tập, chúng em đã nhận được những chia sẻ kiến thức, và sự giúp đỡ quý giá từ thầy và các anh chị bạn trong lớp học. Đây thật sự là một trải nghiệm tuyệt vời đối với chúng em.

Xin cảm ơn thầy vì tất cả. Chúng em xin chúc thầy tiếp tục nhiệt huyết, và thành công hơn trong công tác giảng dạy và nghiên cứu khoa học. Xin chúc các anh chị trong lớp hoàn thành tốt đề tài luận văn tốt nghiệp thạc sĩ khoa học.

Tp. Hồ Chí Minh, tháng 12 năm 2023

Lê Nhựt Nam

We never know how high we are
Till we are called to rise;
And then, if we are true to plan,
Our statures touch the skies
The Heroism we recite
Would be a daily thing,
Did not ourselves the Cubits warp
For fear to be a King

EMILY DICKINSON (1830 – 1886)

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

$\mathcal{R}$	Ngưỡng lợi ích tối thiểu
CPU	Bộ xử lý trung tâm (Central processing unit)
CSP	Bài toán thỏa mãn ràng buộc (Constraints satisfaction problem)
GPU	Bộ xử lý đồ họa (Graphic processing unit)
HTWUI	Itemset TWU cao (High-TWU itemset)
HUI	Itemset lợi ích cao (High-utility itemset)
HUPM	Khai thác mẫu lợi ích cao (High-utility pattern mining)
ILP	Quy hoạch số nguyên tuyến tính (Integer linear programming)
IP	Quy hoạch số nguyên (Integer programming)
LP	Quy hoạch tuyến tính (Linear programming)
LP relaxation	Lời giải nới lỏng của bài toán quy hoạch số nguyên
LUI	Itemset lợi ích thấp (Low-utility itemset)
SHUI	Itemset lợi ích cao nhạy cảm (Sensitive high-utility itemset)

NSHUI	Itemset lợi ích cao không nhạy cảm (Non-sensitive high-utility itemset)
PPFIM	Bảo vệ tính riêng tư trong khai thác tập phổ biến Privacy-preserving frequent itemset mining
PPUM	Bảo vệ tính riêng tư trong khai thác mẫu hữu ích (Privacy-preserving utility mining)
SIP	Tỷ lệ thông tin nhạy cảm (Sensitive information percentage)

DANH MỤC CÁC THUẬT NGỮ

Augmented form	Dạng tăng cường (bài toán quy hoạch tuyến tính)
Constraint	Ràng buộc
Diet problem	bài toán ăn kiêng
Duality	Đối ngẫu
Ellipsoid method	Phương pháp ellipsoid
Fourier–Motzkin elimination	Phép khử Fourier–Motzkin
Linear programming (LP)	Quy hoạch tuyến tính
Mixed integer programming (MIP)	Quy hoạch nguyên hỗn hợp
Manufacturing schedules (MS)	Lập lịch sản suất
Maximize	Cực đại hóa
Minimize	Cực tiểu hóa
Matroid theory	Lý thuyết matroid
Operation research	Vận trù học
Optimization problem	Bài toán tối ưu hóa
Objective function	Hàm mục tiêu
Polyhedron	Đa diện
Programme	Quy hoạch
Primal	Bài toán gốc
Resource allocation (RA)	Phân phối tài nguyên
Integer programming (IP)	Quy hoạch nguyên
Interior point method	Phương pháp điểm trong
System of linear inequalities	Hệ bất đẳng thức tuyến tính
Standard form	Dạng chuẩn (bài toán quy hoạch tuyến tính)
Slack variables	Biến chùng (biến bù)

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ

CHƯƠNG 1

MỞ ĐẦU

Trong chương này, chúng tôi trình bày động lực nghiên cứu về quy hoạch tuyến tính và các ứng dụng của nó trong nghiên cứu lý thuyết cũng như thực tiễn. Bên cạnh đó, chúng tôi cũng trình bày tổng quan về lịch sử hình thành và phát triển của lĩnh vực nghiên cứu Toán học này.

1.1. Tổng quan về đề tài

1.1.1. Động lực nghiên cứu

Trong cuộc sống con người, có nhiều bài toán như lập lịch công việc hay canh tác cây trồng nông nghiệp có thể hình thức hóa thành bài toán tối ưu. Bài toán này cực đại hoặc cực tiểu một hàm dưới (các) ràng buộc bất đẳng thức/ đẳng thức, và thường có dạng như sau:

$$\begin{aligned} &\min / \max f(x) \\ &\text{s.t. } g(x) \end{aligned} \tag{1.1}$$

Trong đó: $f(x)$ là hàm mục tiêu, $g(x)$ là ràng buộc (bất đẳng thức/ đẳng thức)

Và trong trường hợp đơn giản nhất, hàm mục tiêu là hàm tuyến tính và các ràng buộc là ràng buộc tuyến tính. Đây là bài toán quy hoạch tuyến tính hay tối ưu tuyến tính.

1.1.2. Ý nghĩa khoa học - thực tiễn ứng dụng

Quy hoạch tuyến tính được sử dụng nhiều trong lĩnh vực tối ưu hóa. Nhiều vấn đề trong nghiên cứu vận trù học cũng được khai triển về dạng bài toán quy

hoạch tuyến tính để tăng tốc thời gian giải và giảm độ phức tạp bài toán. Hơn nữa, trong các lĩnh vực nghiên cứu khác như lý thuyết đồ thị cũng tận dụng kỹ thuật này để giải quyết một số bài toán như luồng trong mạng.

Trong ứng dụng thực tiễn, quy hoạch tuyến tính được sử dụng để giải quyết nhiều bài toán quản lý cho doanh nghiệp như lập lịch, hay vận chuyển nguyên liệu. Cụ thể, Google sử dụng kỹ thuật này để ổn định đề xuất cho các videos Youtube.

1.1.3. Những đóng góp của tiểu luận

1.1.4. Cấu trúc của tiểu luận

Nội dung nghiên cứu của tiểu luận được trình bày trong bảy chương:

- Chương 1 - Mở đầu
- Chương 2 - Mô hình bài toán
- Chương 3 - Phương pháp hình học cho bài toán quy hoạch tuyến tính hai biến
- Chương 4 - Phương pháp đơn hình
- Chương 5 - Phương pháp lập trình cho quy hoạch tuyến tính
- Chương 6 - Một số ứng dụng của quy hoạch tuyến tính
- Chương 7 - Kết luận và hướng nghiên cứu tương lai

1.2. Lịch sự hình thành và phát triển

> "A certain wide class of practical problems appears to be just beyond the range of modern computing machinery. These problems occur in everyday life; they run the gamut from some very simple situations that confront an individual to those connected with the national economy as a whole. Typically, these problems involve a complex of different activities in which one wishes to know which activities to emphasize in order to carry out desired objectives under known limitations."
>
> Tạm dịch: Một lớp các bài toán thực tế cụ thể nhất định nằm ngoài khả năng tính toán của máy tính hiện đại. Những bài toán này xuất hiện mọi lúc trong đời sống hằng ngày; chúng có thể nằm trong một số tình huống hết sức đơn giản hoặc thậm chí là liên quan đến toàn bộ nền kinh tế quốc gia. Thông thường, những bài toán này phát sinh một chuỗi phức tạp các mục tiêu khác nhau mà trong đó người ta mong muốn biết cần tối đại mục tiêu nào trong những giới hạn cho trước.
>
> George B. Dantzig, 1948

1.2.1. Trước những năm 1930

Vào những năm 1827, Jean-Baptiste Joseph Fourier đã công bố phương pháp giải hệ bất đẳng thức tuyến tính - một trong những vấn đề toán học đã có từ rất lâu về trước. Sau này, Theodore Motzkin đã khám phá lại phương pháp và phát triển thuật toán mà ngay nay ta biết đến với tên gọi phép khử Fourier–Motzkin.

Trong những năm 1930, nhà toán học Liên Xô Leonid Kantorovich và nhà kinh tế học người Mỹ Tjalling Charles Koopmans đã độc lập nghiên cứu các ứng dụng của quy hoạch tuyến tính. Họ đã cùng nhau đưa LP lên một tầm cao mới bằng cách chứng minh những ứng dụng rộng rãi của nó trong các vấn đề phân phối tài nguyên, và lập lịch sản xuất. Sau này, họ cùng nhau nhận giải Nobel

Kinh tế vào năm 1975.

Năm 1941, Frank Lauren Hitchcock cũng đã thành công trong việc xây dựng các bài toán vận tải như quy hoạch tuyến tính và đưa ra phương án giải quyết tương tự như phương pháp đơn hình sau này. Tuy nhiên, ông đã qua đời và giải Nobel không thể trao cho ông.

1.2.2. Thời kỳ sơ khai (1947 - 1990)

Trong những năm 1946 - 1947, nhà toán học người Mỹ George Bernard Dantzig[1] đã làm việc độc lập để xây dựng phương pháp quy hoạch tuyến tính tổng quát trong bài toán lập lịch bay cho không quân Hoa Kỳ. Vào năm 1947, ông đã đề xuất phương pháp đơn hình - phương pháp có thể giải quyết hầu hết các bài toán mà hình thức hóa thành dạng bài toán quy hoạch tuyến tính. Điều này về cơ bản đã cách mạng hóa việc sử dụng quy hoạch tuyến tính trong thực tế.

Sau đó, vào mùa thu năm 1947, Jack Laderman thuộc Dự án Mathematical Tables Project of the National Bureau of Standards đã sử dụng phương pháp đơn hình trong việc giải bài toán ăn kiêng[2] và kiểm chứng mô hình của George Stigler. Đây là dự án tính toán quy mô lớn đầu tiên của lĩnh vực tối ưu hóa, bài toán quy hoạch tuyến tính gồm 9 phương trình với 77 biến ẩn. Trong quá trình thực hiện, dự án cần chín nhân viên sử dụng máy tính để bàn vận hành bằng tay và mất 120 ngày công để tìm ra giải pháp tối ưu là 39,69 USD. Dự đoán của Stigler chỉ sai lệch 0,24 USD mỗi năm!

Vào năm 1951, mã nguồn máy tính cho quy hoạch tuyến tính lần đầu tiên được công bố bởi National Bureau of Standards (bây giờ được gọi là NIST) được

[1]Dantzig được biết đến với việc giải quyết hai vấn đề mở chưa có lời giải trong lĩnh vực thống kê, nhầm chúng với bài tập về nhà sau khi đến giảng muộn tại trường Đại học Berkeley.

[2]Bài toán ăn kiêng là một trong những bài toán tối ưu hóa đầu tiên được nghiên cứu vào những năm 1930 và 1940. Vấn đề được thúc đẩy bởi mong muốn của Quân đội nhằm giảm thiểu chi phí cho lính GI ăn tại thực địa trong khi vẫn cung cấp chế độ ăn uống lành mạnh. Một trong những nhà nghiên cứu đầu tiên nghiên cứu vấn đề này là George Stigler, người đã đưa ra phỏng đoán có căn cứ về giải pháp tối ưu bằng phương pháp heuristic. Dự đoán của ông về chi phí cho một chế độ ăn tối ưu là 39,93 USD mỗi năm (giá năm 1939).

chạy trên máy tính SEAC. Với một bài toán LP 48 phương trình và 71 biến ẩn, máy tính cần 18 giờ và 73 lần lặp đơn hình để có thể đưa ra lời giải tối ưu. Đến năm 1954, Card Programmable Calculator được tạo ra. Thời gian để giải một bài toán LP với 26 phương trình và 71 biến ẩn giảm xuống còn 8 tiếng.

Một cách tóm tắt sự phát triển trong những năm 1955 - 1973:

- 1954-55: IBM 701, 100-200 rows
- 1956: IBM 704, 4 K "core", RSLP1, 255 rows
- 1962-66: 7090/94, LP/90/94, 1024 rows
- 1966-70: IBM 360, MPS/360 & MPSX/370 giải hệ LP thực đầu tiên.
- 1971-73: MPS III/Whizard, 32000 rows

Trong thập kỷ 70, sự quan tâm dành cho nghiên cứu và ứng dụng của tối ưu hóa trở nên lớn. Điều này được thể hiện ở việc nhiều các ứng dụng lập lịch ở quy mô lớn được phát triển và tài nguyên đầu tư nghiên cứu trở nên nhiều hơn. Tuy nhiên, nhiều khó khăn và thách thức đã xuất hiện:

- Chi phí xây dựng hệ thống ứng dụng rất tốn kém và đầy rủi ro.
- Chu kỳ phát triển dự án rất dài, từ 3-4 chu kỳ.
- Các nhà lập trình và chủ sở hữu đối mặt với nhiều vấn đề chuyên môn như: máy tính, dữ liệu, thuật toán và kỹ thuật mô hình hóa.
- Công nghệ tính toán chưa sẵn sàng.

Thế nên, sự phát triển của LP (MIP cũng gặp tình trạng tương tự) gặp trở ngại, kéo theo đó là sự vỡ mộng và phần lớn sự vỡ mộng đó vẫn còn tồn tại cho đến ngày nay.

Đến giữa thập kỷ 80, người ta cho rằng LP đã phát triển đến hết mức có thể, đỉnh cao nhất thời điểm hiện tại có lẽ là MPSX/370 và MPSIII. Nhưng LP chắc

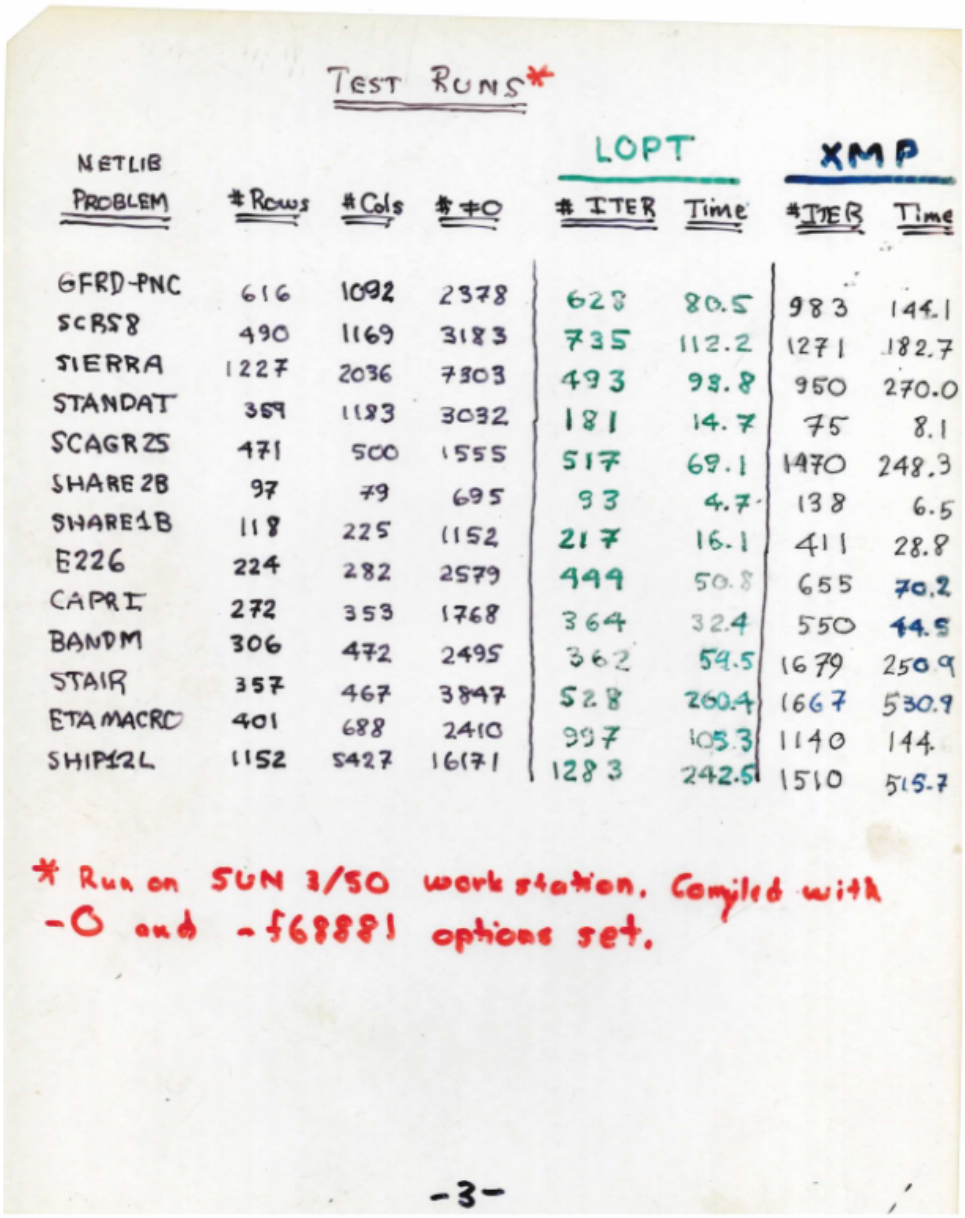
chắn không phải là một vấn đề được giải quyết... ví dụ: Mô hình LP của hãng hàng không "Không thể giải quyết được" với 4420 ràng buộc, 6711 biến. Tuy vậy, trong thời gian này cũng có nhiều sự phát triển đáng chú ý:

- Máy tính IBM được giới thiệu vào năm 1981.
- Cơ sở dữ liệu quan hệ được phát triển.
- Chứng minh được LP có thể giải được trong thời gian đa thức.
  - Năm 1979, Khachiyan chỉ ra rằng LP có thể giải được trong thời gian đa thức bằng "phương pháp ellipsoid". Đây là một bước đột phá về mặt lý thuyết hơn là một bước đột phá thực tế, vì trong thực tế, thuật toán khá chậm.
  - Năm 1984, Karmarkar đã phát triển "phương pháp điểm trong", một thuật toán thời gian đa thức khác cho LP, cũng hiệu quả trong thực tế. Cùng với phương pháp đơn hình, đây là phương pháp được lựa chọn ngày nay để giải LP.

Trong thời gian này, lý thuyết vận trù học có nhiều bước phát triển, đặc biệt là lý thuyết matroid. Từ những năm 1983, ngay sau khi công bố giới thiệu máy tính, IBM bắt đầu phát triển mã nguồn LP. Vào năm 1985 - 1987, Tom Baker đã giới thiệu mã nguồn LP cho sản phẩm Chesapeake Decision Sciences MIMI. Và mãi đến 1988, IBM CPLEX được thành lập.

Cho đến cuối thập kỷ 80 này, LP vẫn chưa thoát khỏi bế tắc. Một số bài toán quy hoạch tuyến tính trở nên rất khó giải, ngay cả đối với các bộ giải LP thương mại được đánh giá cao như MPSX của IBM. Các bài toán LP có 1500 phương trình và 2500 biến và mất gần 3 giờ để giải nếu chúng giải được. Và kèm theo đó là sự thoái hóa, hội tụ chậm và tính không linh hoạt của hệ thống vẫn là những trở ngại lớn trong các bộ giải tối ưu thương mại.

Đến những năm của thập kỷ 90, nhiều bước tiến mới đã giúp LP cải thiện đáng kể về mặt hiệu năng và mở rộng thêm nhiều hướng ứng dụng mới. Các



TEST RUNS*

NETLIB PROBLEM	#Rows	#Cols	#≠0	LOPT # ITER	LOPT Time	XMP #ITER	XMP Time
GFRD-PNC	616	1092	2378	623	80.5	983	144.1
SCRS8	490	1169	3183	735	112.2	1271	182.7
SIERRA	1227	2036	7303	493	98.8	950	270.0
STANDAT	359	1183	3032	181	14.7	75	8.1
SCAGR25	471	500	1555	517	69.1	1470	248.3
SHARE2B	97	79	695	93	4.7	138	6.5
SHARE1B	118	225	1152	217	16.1	411	28.8
E226	224	282	2579	449	50.8	655	70.2
CAPRI	272	353	1768	364	32.4	550	44.8
BANDM	306	472	2495	362	54.5	1679	250.9
STAIR	357	467	3847	528	260.4	1667	530.9
ETAMACRO	401	688	2410	997	105.3	1140	144.
SHIP12L	1152	5427	16171	1283	242.5	1510	515.7

* Run on SUN 3/50 work station. Compiled with -O and -f6888! options set.

-3-

Figure 1.1: *Kết quả bộ giải CPLEX vào năm 1986.*

phương pháp primal-dual log-barrier đã nâng cao tốc độ giải LP. Bên cạnh đó dữ liệu cũng trở nên đa dạng và dễ dàng truy cập được. Thế nên có nhiều ứng dụng mới từ các bài toán thực tế bắt đầu cho thấy tính khả thi của việc ứng dụng quy hoạch tuyến tính và quy hoạch nguyên như hàng không, hay chuỗi cung ứng.

1.2.3. Những bước phát triển mới

AA Challenge,

AA & US Air Merger,

LAU2

Là một bài toán phân công đội tàu

1.2.4. Quy hoạch tuyến tính ngày nay

Mô hình lập lịch thương mại

Bài toán lập lịch là bài toán quan trọng và được quan tâm bởi doanh nghiệp. Trong thương mại, các bộ giải LP đối mặt với bài toán với 401640 ràng buộc, 1584000 biến. Dưới đây là đánh giá với bộ giải CPLEX qua các năm.

Năm	Bộ giải	Thời gian	Tốc độ cải thiện
1988	CPLEX 1.0	29.8 ngày	1x
1997	CPLEX 5.0	1.5 giờ	480x
2003	CPLEX 9.0	59.1 giây	43500x

Table 1.1: *Test.*

Tiến triển trong LP giai đoạn 2004 - 2015

CHƯƠNG 2

MÔ HÌNH BÀI TOÁN

Trong chương này, chúng tôi trình bày các dạng toán học phục vụ cho việc mô tả một bài toán quy hoạch tuyến tính, bao gồm: dạng chuẩn, và dạng tăng cường. Với các dạng mô hình, chúng tôi trình bày các một số ví dụ minh họa tương ứng. Hơn nữa, chúng tôi cũng điểm qua tính đối ngẫu của bài toán LP.

2.1. Dạng chuẩn LP

Bài toán quy hoạch tuyến tính là bài toán tối ưu hóa mà bao gồm ba thành phần chính

- Cực đại hóa (hoặc cực tiểu hóa) một hàm mục tiêu tuyến tính (hoặc hàm affine).
- Hàm mục tiêu có n biến quyết định.
- Thỏa mãn một tập các ràng buộc mà được khai triển bởi các đẳng thức hoặc bất đẳng tuyến tính.

Dạng chuẩn của một bài toán LP như sau:

$$\begin{aligned} \text{maximize/ minimize} \quad & c_1x_1 + \cdots + c_nx_n \\ \text{subject to} \quad & a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ & a_{21}x_1 + \cdots + a_{2n}x_n = b_2 \\ & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \\ & x_1, x_2, ..., x_n \geq 0 \end{aligned} \tag{2.1}$$

Trong đó: b_i, c_i và a_{ij} là các hằng số cố định, và x_i là các biến số thực cần được quyết định. Giả định rằng mỗi phương trình đã được nhân với trừ đơn vị, thế nên $b_i \geq 0$

Trong dạng sử dụng ma trận, bài toán dạng chuẩn có thể viết một cách gọn hơn như sau:

$$\begin{aligned} \text{maximize/ minimize} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{2.2}$$

Trong đó: vector quyết định $\mathbf{x} \in \mathbb{R}^n$, vector hệ số dữ liệu mục tiêu $\mathbf{c}^\top \in \mathbb{R}^n$, ma trận dữ liệu ràng buộc $\mathbf{A} \in \mathbb{R}^{m \times n}$, và vector dữ liệu bên phải $\mathbf{b} \in \mathbb{R}^m$. Ràng buộc bất đẳng thức $\mathbf{x} \geq \mathbf{0}$ ám chỉ mỗi thành phần của vector $\mathbf{x}$ là không âm.

2.2. Các dạng biến thể của LP

Trong thực tế, các bài toán có thể được mô hình hóa thành những dạng khác nhau. Chúng tôi điểm qua những biến thể của quy hoạch tuyến tính mà có thể chuyển đổi về thành dạng chuẩn, bao gồm: dạng biến chùng, biến thừa, và biến tự do.

2.2.1. Dạng biến chùng

Ta xem xét bài toán LP trong trường hợp tập ràng buộc là các bất đẳng thức tuyến tính như sau:

$$\begin{aligned} \text{maximize/ minimize} \quad & c_1 x_1 + \cdots + c_n x_n \\ \text{subject to} \quad & a_{11} x_1 + \cdots + a_{1n} x_n \leq b_1 \\ & a_{21} x_1 + \cdots + a_{2n} x_n \leq b_2 \\ & \vdots \\ & a_{m1} x_1 + \cdots + a_{mn} x_n \leq b_m \\ & x_1, x_2, ..., x_n \geq 0 \end{aligned} \tag{2.3}$$

Bằng cách thêm vào bài toán các biến không âm $x_{n+i}, i = 1...m$ để chuyển tập ràng buộc bất đẳng thức thành tập ràng buộc đẳng thức. Các biến này được gọi là các biến chùng:

$$\begin{aligned} \text{maximize/ minimize} \quad & c_1x_1 + \cdots + c_nx_n \\ & a_{11}x_1 + \cdots + a_{1n}x_n + x_{n+1} = b_1 \\ & a_{21}x_1 + \cdots + a_{2n}x_n + x_{n+2} = b_2 \\ \text{subject to} \quad & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n + x_{n+m} = b_m \\ & x_1, x_2, ..., x_n, x_{n+1}, ..., x_{n+m} \geq 0 \end{aligned} \tag{2.4}$$

2.2.2. Dạng biến thừa

Nếu ta đổi dấu các bất đẳng thức ràng buộc ở bài toán 2.3, ta được:

$$\begin{aligned} \text{maximize/ minimize} \quad & c_1x_1 + \cdots + c_nx_n \\ & a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \\ & a_{21}x_1 + \cdots + a_{2n}x_n \geq b_2 \\ \text{subject to} \quad & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_m \\ & x_1, x_2, ..., x_n \geq 0 \end{aligned} \tag{2.5}$$

Xem xét một bất đẳng thức cụ thể thứ i

$$a_{i1}x_1 + \cdots + a_{in}x_n \geq b_i \tag{2.6}$$

Dễ dàng biến đổi bất đẳng thức trên thành đẳng thức bằng cách:

$$a_{i1}x_1 + \cdots + a_{in}x_n - y_i = b_i \tag{2.7}$$

Từ đó, ta dễ dàng chuyển đổi về dạng chuẩn LP. Những biến y_i được gọi là biến thừa.

2.2.3. Dạng biến tự do

Nếu một bài toán LP cho trước ở dạng chuẩn mà có một hay nhiều biến ẩn không có ràng buộc dương, thì bài toán có thể được biến đổi về dạng chuẩn bằng một trong hai cách.

- Cách 1: Biến đổi biến tự do dựa trên ràng buộc và thay thế vào hàm mục tiêu và các ràng buộc còn lại.
- Cách 2: Thêm biến vào bài toán.

2.2.4. Ví dụ cụ thể

Xem xét bài toán:

$$\begin{aligned} \text{minimize} \quad & x_1 + 2x_2 + 3x_3 \\ \text{subject to} \quad & x_1 + x_2 + 2x_3 = 4 \\ & 3x_1 + 4x_2 + 5x_3 = 9 \\ & x_2, x_3 \geq 0 \end{aligned} \tag{2.8}$$

Bởi vì x_1 là biến tự do, dựa vào ràng buộc thứ nhất, ta có:

$$x_1 = 4 - x_2 - 2x_3 \tag{2.9}$$

Thay vào hàm mục tiêu và ràng buộc thứ hai, ta thu được bài toán tương đương như sau:

$$\begin{aligned} \text{minimize} \quad & 4 + x_2 + x_3 \\ \text{subject to} \quad & -x_2 + x_3 = 3 \\ & x_2, x_3 \geq 0 \end{aligned} \tag{2.10}$$

2.3. Tính đối ngẫu

Mọi bài toán quy hoạch tuyến tính, được xem như bài toán gốc (primal), có thể được biến đổi thành một bài toán đối ngẫu mà cho ta một chặn trên đối với giá trị tối ưu của bài toán gốc. Nếu xem xét dạng ma trận, ta có thể khai triển bài toán primal như sau:

2.4. Các ví dụ thực tế về bài toán LP

LP được xem như một mô hình hiệu quả trong việc giải quyết nhiều bài toán thực tế liên quan đến các hiện tượng phân chia công việc và kinh tế học.

2.4.1. Bài toán ăn kiêng - Diet Problem

Làm thế nào chúng ta có thể xác định được chế độ ăn uống tiết kiệm nhất, đáp ứng được các yêu cầu dinh dưỡng tối thiểu cơ bản để có sức khỏe tốt? Đó là một bài toán khó, nhất là với một chuyên gia dinh dưỡng trong quân đội. Và thật vậy, bài toán này là một trong những bài toán tối ưu hóa đầu tiên được nghiên cứu vào những năm 1930 và 1940. Vấn đề được thúc đẩy bởi mong muốn của Quân đội nhằm giảm thiểu chi phí cho lính GI ăn tại thực địa trong khi vẫn cung cấp chế độ ăn uống lành mạnh. Một trong những nhà nghiên cứu đầu tiên nghiên cứu vấn đề này là George Stigler, người đã đưa ra phỏng đoán có căn cứ về giải pháp tối ưu bằng phương pháp heuristic. Dự đoán của ông về chi phí cho một chế độ ăn tối ưu là 39,93 USD mỗi năm (giá năm 1939).

Để minh họa, chúng ta sẽ xem xét bài toán dưới góc độ đơn giản. Giả định rằng cho sẵn trong siêu thị n thức ăn khác nhau, thức ăn thứ j có mức giá c_j trên một đơn vị. Ngoài ra còn có m thành phần dinh dưỡng cơ bản và để đạt được một chế độ ăn lành mạnh (balance diet), mỗi cá nhân cần phải nhận được ít nhất b_i đơn vị của thành phần dinh dưỡng thứ i trên một ngày. Cuối cùng, giả định rằng mỗi đơn vị thức ăn chứa a_{ij} đơn vị của thành phần dinh dưỡng thứ i. Ta có thể mô tả bài toán dưới dạng công thức như sau:

$$\begin{aligned} \text{minimize} \quad & c_1x_1 + c_2x_2 + ... + c_nx_n \\ \text{subject to} \quad & a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i, i = 1...m \\ & x_1, x_2, ..., x_n \geq 0 \end{aligned} \tag{2.11}$$

2.4.2. Bài toán phân phối tài nguyên - Resource-Allocation Problem

Giả sử chúng ta sở hữu một cơ sở có khả năng sản xuất n sản phẩm khác nhau, mỗi sản phẩm có thể yêu cầu lượng m tài nguyên khác nhau. Mỗi sản phẩm cần được sản xuất ở bất kỳ mức độ $x_j \geq 0, j = 1, .., n$ nào, và mỗi đơn vị của sản phẩm thứ j có thể được bán với giá π_j và cần có a_{ij} đơn vị của tài nguyên thứ $i, i = 1, ..., m$. Giả định rằng việc sản xuất sản phẩm có tính tuyến tính, nếu cho trước một tập gồm m số $b_1, b_2, ..., b_m$ thể hiện định lượng sẵn có của m nguồn tài nguyên, và ta mong muốn việc sản xuất sản phẩm đạt doanh thu tối đa. Bài toán này là một bài toán quy hoạch tuyến tính:

$$\begin{aligned} \text{minimize} \quad & \pi_1x_1 + \pi_2x_2 + ... + \pi_nx_n \\ \text{subject to} \quad & a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i, i = 1...m \\ & x_1, x_2, ..., x_n \geq 0 \end{aligned} \tag{2.12}$$

2.4.3. Bài toán vận chuyển - Transportation Problem

Mục tiêu của bài toán vận chuyển là giảm thiểu chi phí vận chuyển của một mặt hàng nhất định từ nguồn hoặc nơi xuất xứ (ví dụ: nhà máy) đến điểm đích (ví dụ: kho hàng, khách hàng cuối cùng), thỏa mãn tất cả các ràng buộc được yêu cầu không chỉ bởi nơi xuất xứ (ví dụ: tối đa số lượng hàng hóa có thể được gửi từ đó) mà còn theo điểm đến (ví dụ: số lượng sản phẩm tối thiểu cần được vận chuyển đến kho, số lượng sản phẩm theo yêu cầu của mỗi khách hàng). Ngoài những ràng buộc này, cũng cần xem xét chi phí vận chuyển hàng hóa từ nơi đi đến nơi đến (chi phí vận chuyển).

Trước hết, gọi các định lượng $a_1, a_2, ..., a_m$ của một sản phẩm cụ thể cần được

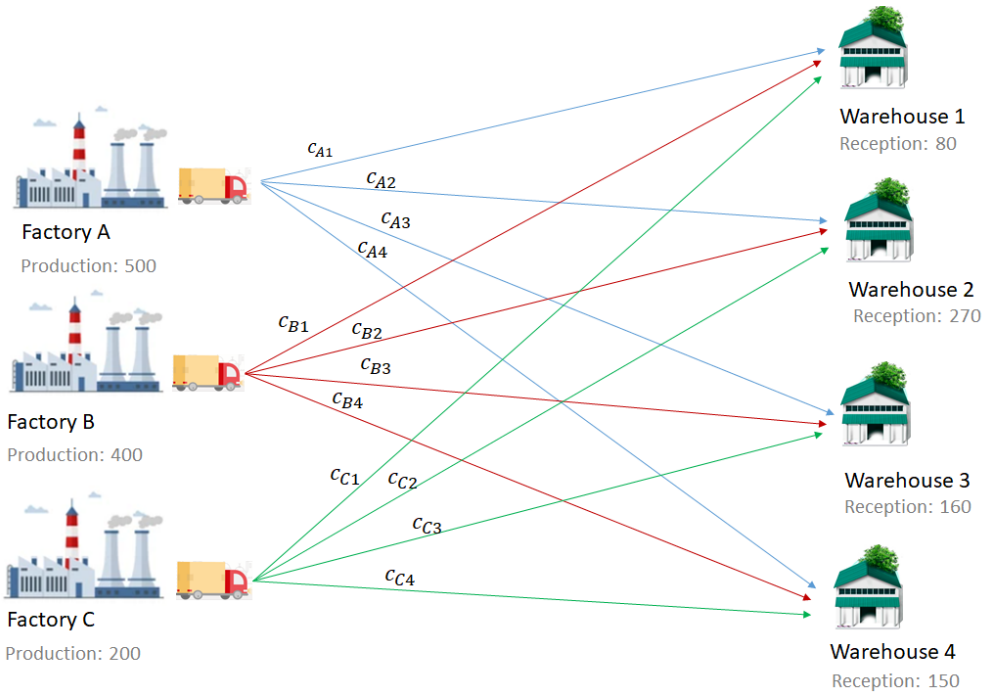


Figure 2.1: *Minh họa về bài toán vận chuyển.*

gửi đi từ mỗi một m vị trí và nhận với số lượng $b_1, b_2, ..., b_n$ tương ứng tại mỗi n điểm đích. Với mỗi việc giao sản phẩm, một đơn vị sản phẩm từ vị trí nguồn i đến vị trí đích j tiêu hao chi phí c_{ij}. Gọi x_{ij} là số sản phẩm giao giữa mỗi cặp vị trí nguồn và đích, $i = 1...m$ và $j = 1...n$; cần phải tìm x_{ij} để thỏa mãn yêu cầu vận chuyển và cực tiểu tổng chi phí của công việc vận chuyển. Và ta có bài toán quy hoạch tuyến tính như sau:

$$\begin{aligned} \text{minimize} \quad & \sum_{ij} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = a_i, i = 1, ...m \\ & \sum_{i=1}^{m} x_{ij} = b_j, j = 1, ...n \\ & x_{ij} \geq 0, i = 1, ...m; j = 1, ...n \end{aligned} \tag{2.13}$$

2.4.4. Bài toán luồng cực đại - Maximal Flow Problem

Có một số bài toán thực tế có thể được mô hình hóa dưới dạng các luồng trong đồ thị đặc biệt gọi là luồng trên mạng. Luồng trên mạng là một đồ thị có hướng có các cạnh được dán nhãn bằng các số không âm biểu thị thông lượng của một loại dòng chảy nào đó: năng lượng điện, hàng hóa sản xuất sẽ được phân phối hoặc phân phối nước thành phố. Hình 2.3 là một ví dụ trừu tượng về luồng trên mạng.

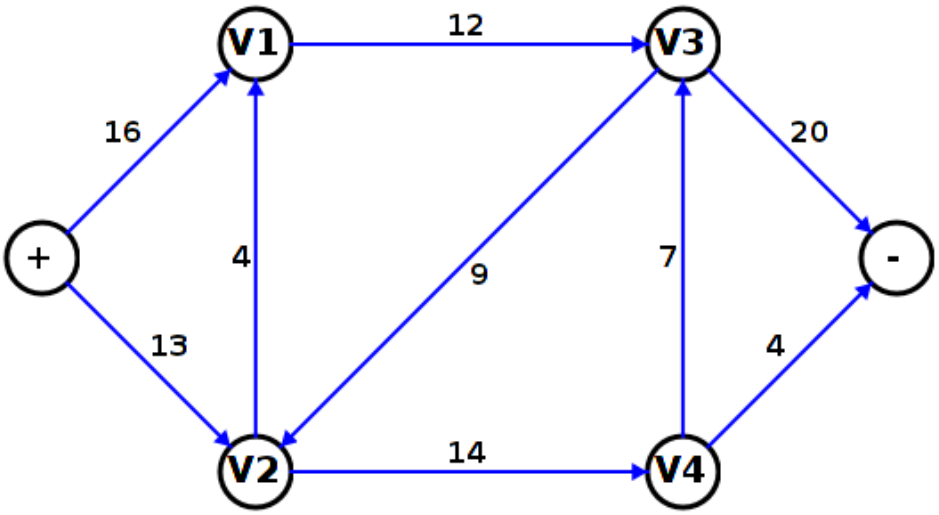


Figure 2.2: *Minh họa về một luồng trên mạng.*

Mạng luồng có thể có các đỉnh đặc biệt gọi là nguồn (sources) và đích (sinks).

- Một nguồn tạo ra hoặc tạo ra bất cứ thứ gì đang chảy trong mạng. Đỉnh "+" thể hiện điểm nguồn trong luồng trên mạng trong Hình 2.3. Tùy thuộc vào bản chất chính xác của vấn đề, nguồn cũng có thể có giới hạn hoặc công suất.

- Một đích tiêu thụ bất cứ thứ gì đang chảy trong mạng. Đỉnh "-" thể hiện điểm đích trong luồng trên mạng trong Hình 2.3. Tùy thuộc vào bản chất chính xác của vấn đề, điểm đích cũng có thể có giới hạn hoặc dung tích.

Nút nguồn và nút đích của một mạng là phân biệt nhau, ta gọi chúng lần

lượt là nút 1 và nút m. Tất cả các nút còn lại phải thỏa mãn điều kiện rằng lưu lượng vào chúng bằng 0. Tuy nhiên, nút nguồn có một luồng lưu lượng ra và nút đích có một luồng lưu lượng vào. Luồng lưu lượng ra f của nút nguồn sẽ bằng luồng lưu lượng vào của nút đích như một hệ quả vì điều kiện của các nút còn lại.

Một tập hợp các luồng cung thỏa mãn các điều kiện này được gọi là một luồng trong mạng có giá trị f. Bài toán luồng cực đại là vấn đề xác định luồng cực đại có thể được thiết lập trong mạng như vậy.

$$\begin{aligned} \text{maximize} \quad & f \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{1j} - \sum_{j=1}^{n} x_{j1} - f = 0 \\ & \sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = 0, i \neq 1, m \\ & \sum_{j=1}^{n} x_{mj} - \sum_{j=1}^{n} x_{jm} + f = 0 \\ & 0 \leq x_{ij} \leq k_{ij}, \forall i, j \end{aligned} \tag{2.14}$$

2.4.5. Bài toán chuỗi cung ứng - Supply-Chain Problem

2.4.6. Bài toán tối ưu phân lớp tuyến tính và Support Vector Machine

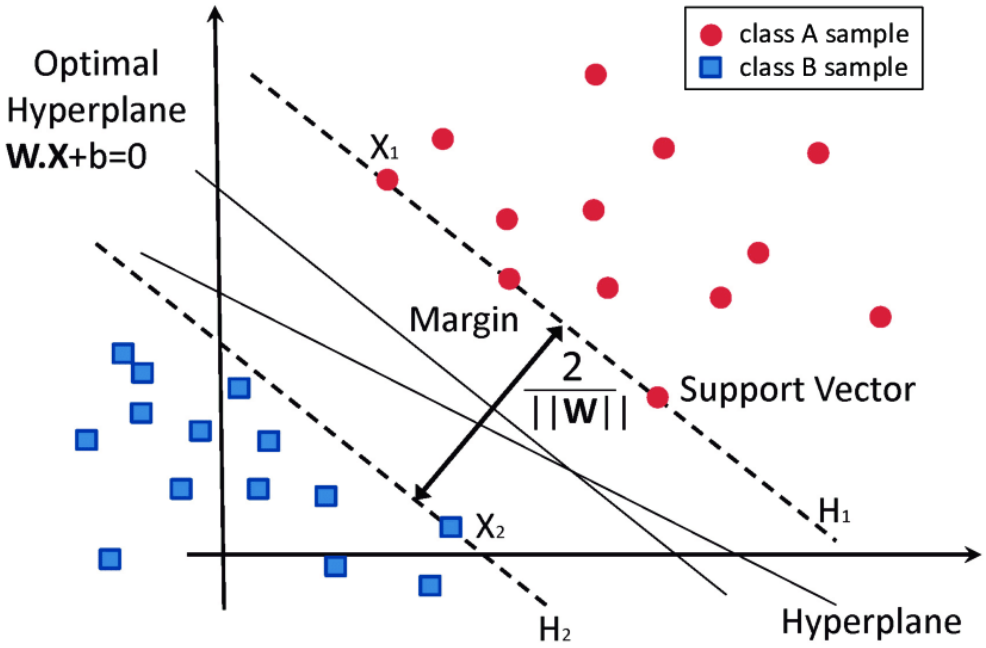


Figure 2.3: *Minh họa về support vector machine.*

CHƯƠNG 3

PHƯƠNG PHÁP HÌNH HỌC CHO BÀI TOÁN QUY HOẠCH TUYẾN TÍNH HAI BIẾN

Trong chương này, chúng tôi trình bày phương pháp hình học để giải quyết bài toán quy hoạch tuyến tính hai biến thông qua các ví dụ và bài toán thực tế. Bên cạnh đó, chúng tôi cũng trình bày tính hình của LP.

3.1. Phương pháp hình học giải bài toán quy hoạch tuyến tính hai biến

3.2. Bài toán thực tế

3.3. Hình học của LP

CHƯƠNG 4

PHƯƠNG PHÁP ĐƠN HÌNH

Trong chương này, chúng tôi trình bày ý tưởng của thuật toán đơn hình và áp dụng giải quyết bài toán quy hoạch tuyến tính trong không gian hai chiều.

4.1. Ý tưởng thuật toán

4.2. Ví dụ minh họa thuật toán

4.3. Bàn luận về thuật toán

CHƯƠNG 5

PHƯƠNG PHÁP LẬP TRÌNH CHO QUY HOẠCH TUYẾN TÍNH

Trong chương này, chúng tôi trình bày ví dụ minh họa trong việc sử dụng ngôn ngữ lập trình Python và thư viện bộ giải Gurobi[3].

5.1. Cài đặt phần mềm

Trước tiên, chúng ta cần có sẵn môi trường Python. Sau đó, chúng ta sẽ cài đặt thư viện Gurobi. Gurobi là một trong những công cụ giải quyết tối ưu hóa mạnh mẽ và nhanh nhất và công ty liên tục tung ra các tính năng mới. Câu lệnh cài đặt như sau

```
python -m pip install -i https://pypi.gurobi.com gurobipy==10.0
```

5.2. Bài toán quy hoạch tuyến tính

Xem xét một công ty sản xuất sản xuất hai mặt hàng: cốc và đĩa.

- Sau khi làm xong, một chiếc cốc được bán với giá 27 USD và một chiếc đĩa được bán với giá 21 USD.
- Để làm ra mỗi chiếc cốc, chi phí vật liệu là 10 USD và nhân công là 14 USD.
- Để làm ra mỗi chiếc đĩa, chi phí vật liệu là 9 USD và nhân công là 10 USD.
- Để làm ra mỗi chiếc cốc, phải mất 2,2 giờ lao động.

- Để làm ra mỗi chiếc đĩa, phải mất 1 giờ lao động.
- Hãng có nguồn cung nguyên liệu vô hạn.
- Do số lượng công nhân có hạn nên một công ty có tối đa 100 giờ lao động.
- Nhu cầu về cốc là không giới hạn nhưng nhu cầu về đĩa là 30 chiếc.

Mục tiêu của công ty là tối đa hóa lợi nhuận (doanh thu – chi phí).

5.3. Giải bài toán

5.3.1. Bước 1 - Định nghĩa các biến

```
# Create variables
x1 = m.addVar(name="x1")
x2 = m.addVar(name="x2")
```

5.3.2. Bước 2 - Định nghĩa hàm mục tiêu

```
# Set objective
m.setObjective(3*x1 + 2*x2 , GRB.MAXIMIZE)
```

5.3.3. Bước 3 - Định nghĩa các ràng buộc bất đẳng thức

```
# Build (sparse) constraint matrix
m.addConstr(2.2*x1 + x2  <= 100, "c1")
m.addConstr(x2  <= 30, "c3")
m.addConstr(x1  >= 0, "c4")
m.addConstr(x2  >= 0, "c5")
```

5.4. Chương trình hoàn thiện và thực nghiệm

Toàn bộ mã nguồn và kết quả thực nghiệm

lp_demo.py > ...

```python
import gurobipy as gp
from gurobipy import GRB
import numpy as np
import scipy.sparse as sp

try:
    # Create a new model
    m = gp.Model("TTTU_LP_Demo")

    # Create variables
    x1 = m.addVar(name="x1")
    x2 = m.addVar(name="x2")

    # Set objective
    m.setObjective(3*x1 + 2*x2 , GRB.MAXIMIZE)

    # Build (sparse) constraint matrix
    m.addConstr(2.2*x1 + x2  <= 100, "c1")
    m.addConstr(x2  <= 30, "c3")
    m.addConstr(x1  >= 0, "c4")
    m.addConstr(x2  >= 0, "c5")

    # Optimize model
    m.optimize()

    print(f"Optimal solution: ")
    for v in m.getVars():
        print(f"{v.varName} = {v.x}")

except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ": " + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

Figure 5.1: *Mã nguồn chương trình.*

```
(pytorch) ┌─[lnhutnam@namlnhut] - [~/Workspace/Test] - [2023-12-23 07:58:29]
└─[0] <git:(master cf58444*↑) > python lp_demo.py
Set parameter Username
Academic license - for non-commercial use only - expires 2024-01-11
Gurobi Optimizer version 10.0.0 build v10.0.0rc2 (linux64)

CPU model: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 4 rows, 2 columns and 5 nonzeros
Model fingerprint: 0x8dc0c632
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [2e+00, 3e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+01, 1e+02]
Presolve removed 4 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.5545455e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds (0.00 work units)
Optimal objective  1.554545455e+02
Optimal solution:
x1 = 31.818181818181817
x2 = 30.0
(pytorch) ┌─[lnhutnam@namlnhut] - [~/Workspace/Test] - [2023-12-23 08:09:43]
└─[0] <git:(master cf58444*↑) >
```

Figure 5.2: *Kết quả thực nghiệm.*

CHƯƠNG 6

MỘT SỐ ỨNG DỤNG CỦA QUY HOẠCH TUYẾN TÍNH

CHƯƠNG 7

KẾT LUẬN VÀ HƯỚNG NGHIÊN CỨU TƯƠNG LAI

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] Dimitris Bertsimas and John N Tsitsiklis (1997), *Introduction to linear optimization*, vol. 6, Athena scientific Belmont, MA.

[2] Vašek Chvátal (1983), *Linear programming*, Macmillan.

[3] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023, URL: https://www.gurobi.com.

[4] Howard Karloff (2008), *Linear programming*, Springer Science & Business Media.

[5] David G Luenberger, Yinyu Ye, et al. (1984), *Linear and nonlinear programming*, vol. 2, Springer.

[6] Alexander Schrijver (1998), *Theory of linear and integer programming*, John Wiley & Sons.