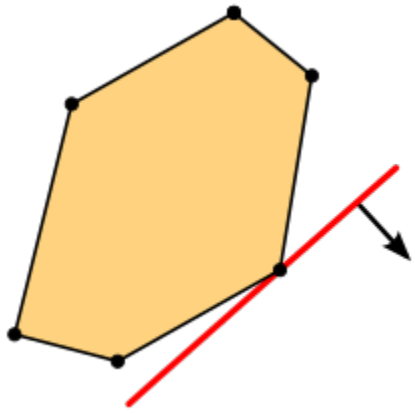
TỐI ƯU HÓA CHI PHÍ
CHỌN THẦU THEO
BẬC SỐ LƯỢNG VÀ CHI PHÍ VẬN TẢI
(SCALE PRICE * TRANSPORTATION COST)

- *Kiến thức được tổng hợp từ internet với sự hỗ trợ của AI – ChatGPT (mode: deep research)*
- *Python code & Excel format – được thiết kế bởi Ths. Nguyễn Minh Ngọc (MSc Suppy Chain & Logistics Management)*
- *ERX Vietnam phát hành*

1. Nguyên lý tối ưu hóa toán học

Trong quản lý chuỗi cung ứng, **bài toán tối ưu hóa chi phí chọn nhà cung cấp** là một dạng cụ thể của bài toán tối ưu hóa toán học. Mục tiêu chung là **tìm phương án đặt hàng từ các nhà cung cấp sao cho thỏa mãn nhu cầu với chi phí thấp nhất**, đồng thời tuân thủ các giới hạn (ví dụ: năng lực cung ứng, ngân sách, v.v.). Đây thực chất là một **bài toán quy hoạch tuyến tính (Linear Programming – LP)** hoặc mở rộng của quy hoạch tuyến tính khi có thêm yếu tố rời rạc (quy hoạch nguyên).

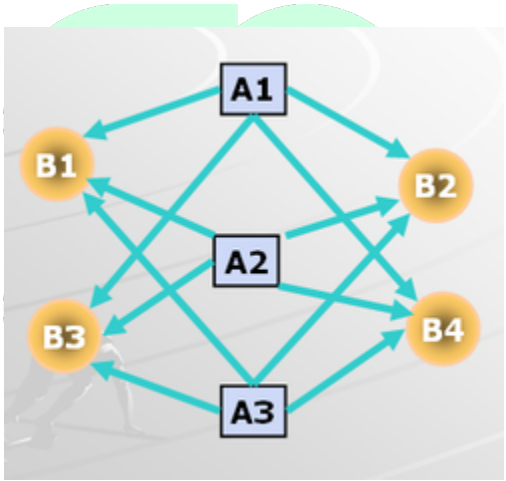


Hình 1: Minh họa hình học một bài toán quy hoạch tuyến tính đơn giản với 2 biến và nhiều ràng buộc bất đẳng thức. Vùng giải pháp hợp lệ (màu vàng) là một đa giác lồi; hàm mục tiêu được biểu diễn bởi đường thẳng màu đỏ và đạt giá trị tối ưu tại biên của vùng (điểm tiếp xúc giữa đường đỏ và đa giác).

(Linear programming - Wikipedia)

Bài toán lựa chọn nhà cung cấp có thể mô hình hóa bằng LP như sau: Giả sử có tập hợp các nhà cung cấp và tập hợp các đơn vị yêu cầu (như nhà máy, kho hoặc cửa hàng). Mỗi nhà cung cấp có chi phí cung ứng (có thể khác nhau cho mỗi đơn vị yêu cầu) và giới hạn về năng lực cung ứng. Mỗi đơn vị yêu cầu có một nhu cầu nhất định cần được thỏa mãn. Hàm mục tiêu thường là **tổng chi phí mua hàng và vận chuyển** – cần được tối thiểu hóa. Các **biến quyết định** có thể là số lượng hàng mua từ nhà cung cấp *i* để phục vụ đơn vị yêu cầu *j*.

Ví dụ kinh điển là **bài toán vận tải** – một trường hợp đặc biệt của LP (Bài toán vận tải – Wikipedia tiếng Việt). Trong bài toán vận tải, ta có *m* điểm phát (nhà cung cấp) với nguồn cung a_i tại mỗi điểm phát *Ai* và *n* điểm thu (đơn vị yêu cầu) với nhu cầu b_j tại mỗi điểm thu *Bj*. Chi phí vận chuyển một đơn vị hàng từ *Ai* đến *Bj* là c_{ij}. Mô hình LP của bài toán vận tải nhằm tìm lượng hàng x_{ij} vận chuyển trên mỗi tuyến *Ai–Bj* sao cho **tổng chi phí vận chuyển** ∑ c_{ij} * x_{ij} là nhỏ nhất, thỏa mãn ràng buộc cung cầu (mỗi *Ai* xuất tổng cộng a_i đơn vị, mỗi *Bj* nhận đủ b_j đơn vị). Bài toán này có thể biểu diễn bằng đồ thị hai phía, trong đó cung từ các *A* phân phối đến các *B*.



Hình 2: Mô hình mạng lưới của bài toán vận tải với 3 điểm phát (A1, A2, A3) và 4 điểm thu (B1, B2, B3, B4). Mỗi cung đường từ một điểm phát đến một item thu có một chi phí nhất định; mục tiêu là vận chuyển sao cho tổng chi phí tối thiểu.

Để giải quyết các bài toán như trên, ta thiết lập **hàm mục tiêu** (total cost) và **các ràng buộc tuyến tính** (nhu cầu, nguồn cung, giới hạn khác). Đối với bài toán chọn nhà cung cấp có **giá bậc thang (volume discounts)**, mô hình cơ bản cần mở rộng thành **quy hoạch tuyến tính nguyên hỗn hợp (MILP)**, vì hàm chi phí không còn tuyến tính đơn thuần mà có các điểm gấp khúc (mức giảm giá thay đổi theo sản lượng mua). Chẳng hạn, một nhà cung cấp có thể áp dụng chính sách *giảm giá theo lũy tiến*: mua 100 sản phẩm đầu tiên giá 20 USD/sản phẩm, mua thêm vượt 100 sẽ giảm còn 18 USD/sản phẩm, v.v. Mô hình toán học khi đó phải sử dụng thêm biến nhị phân để lựa chọn mức giá phù hợp cho mỗi nhà cung cấp.

Tóm lại, nguyên lý chung của tối ưu hóa chi phí chọn thầu gồm:

- **Xác định biến quyết định**: Thông thường là lượng hàng mua từ mỗi nhà cung cấp cho mỗi mặt hàng hoặc địa điểm.
- **Thiết lập hàm mục tiêu**: Tổng chi phí mua và vận chuyển cần tối thiểu hóa.

- **Xây dựng các ràng buộc**: Đáp ứng đầy đủ nhu cầu của mỗi đơn vị; không vượt quá năng lực cung ứng của mỗi nhà cung cấp; và nếu có chiết khấu bậc thang, phải tuân thủ quy luật của các bậc giá.
- **Lựa chọn mô hình phù hợp**: Nếu tất cả quan hệ là tuyến tính, dùng LP thông thường; nếu có yếu tố rời rạc (như chọn mức giá cố định sau khi mua lượng nhất định), dùng MILP với biến nhị phân.

Nhiều nghiên cứu khoa học đã đề cập bài toán này. Chẳng hạn, Ghodsypour và O'Brien (2001) tích hợp các **yếu tố chi phí logistics, nhiều nhà cung cấp và giới hạn công suất** trong bài toán chọn nhà cung cấp và áp dụng mô hình quy hoạch nguyên hỗn hợp để tìm lời giải tối ưu (Model of multiple suppliers). Trước đó, Ghodsypour và O'Brien (1998) đã xây dựng **hệ hỗ trợ quyết định** kết hợp phân tích thứ bậc (AHP) và lập mô hình quy hoạch tuyến tính nhằm chọn nhà cung cấp tối ưu theo nhiều tiêu chí (định tính và định lượng). Những nghiên cứu này đặt nền móng cho việc mô hình hóa lựa chọn nhà cung cấp như một bài toán tối ưu chi phí tổng hợp, xét đồng thời giá mua, chi phí vận chuyển, cũng như các yếu tố hiệu suất cung ứng.

2. Hướng dẫn cơ bản về Python

Python là một ngôn ngữ lập trình thông dụng và mạnh mẽ, rất phù hợp cho các bài toán phân tích dữ liệu và tối ưu hóa. Đối với mục tiêu tối ưu hóa chi phí trong chuỗi cung ứng, Python cung cấp nhiều **thư viện** hữu ích như **Pandas** (xử lý dữ liệu, đặc biệt là đọc/ghi file Excel, CSV), **PuLP** (dựng mô hình tối ưu hóa tuyến tính/nguyên) và nhiều thư viện khác như NumPy, SciPy, etc. Dưới đây là những hướng dẫn cơ bản để bắt đầu:

- **Cài đặt môi trường Python**: Bạn có thể cài Python cùng các thư viện qua phân phối Anaconda, hoặc sử dụng môi trường trực tuyến như **Google Colab** (một dịch vụ cho phép chạy Python trên trình duyệt, không cần cài đặt) hay Jupyter Notebook. Môi trường Jupyter Notebook/Colab rất thuận tiện để viết và chạy từng đoạn mã (code) và xem kết quả tức thì.
- **Các thư viện cần thiết**: Đầu tiên, cài đặt Pandas và PuLP. Nếu dùng Anaconda, các thư viện này có thể đã sẵn có. Nếu dùng Colab, có thể cần chạy !pip install pulp trong notebook để cài PuLP. Pandas thường được cài sẵn trong Colab.
- **Nguyên tắc lập trình cơ bản trong Python**: Hiểu các kiểu dữ liệu chính (số, chuỗi, boolean), cấu trúc dữ liệu như *list* (danh sách), *dict* (từ điển), *tuple*, v.v. Nắm cách viết vòng lặp (for, while), câu lệnh điều kiện (if-elif-else), và định nghĩa hàm (def). Những kiến thức này giúp bạn xử lý linh hoạt dữ liệu đầu vào trước khi đưa vào mô hình tối ưu.

- **Đọc/ghi file Excel với Pandas**: Thư viện Pandas cung cấp hàm pd.read_excel() để đọc dữ liệu từ file Excel vào **DataFrame** (bảng dữ liệu trong Python) (pandas.read_excel — pandas 2.2.3 documentation - PyData |). Tương tự, có thể ghi kết quả ra Excel bằng df.to_excel(). Bạn nên làm quen với việc truy cập các phần tử DataFrame, lọc dữ liệu, và thao tác với bảng (như group, merge nếu cần).
- **Thao tác với PuLP**: PuLP cho phép tạo và giải các bài toán tối ưu một cách trực quan. Các bước cơ bản khi dùng PuLP:
    1. **Khởi tạo mô hình**: Sử dụng LpProblem("Name", LpMinimize) cho bài toán minimization (tối thiểu hóa) hoặc LpMaximize cho maximization.
    2. **Tạo biến quyết định**: Sử dụng LpVariable để tạo biến, có thể là liên tục hoặc nguyên/binary tùy bài toán (tham số lowBound, upBound, cat để giới hạn miền giá trị và kiểu biến).
    3. **Định nghĩa hàm mục tiêu**: Dùng toán tử cộng/trừ giữa các biến để xây dựng biểu thức chi phí tổng, rồi thêm vào mô hình qua model += (biểu_thức).
    4. **Thêm các ràng buộc**: Tương tự, dùng model += (biểu_thức_ràng_buộc) cho mỗi ràng buộc. Biểu thức dạng tuyến tính so sánh (<=, >=, ==) giữa tổ hợp biến và hằng số.
    5. **Giải bài toán**: Gọi model.solve() để chạy bộ giải (mặc định PuLP dùng CBC – một solver mã nguồn mở kèm theo).
    6. **Kiểm tra kết quả**: Dùng value(var) hoặc var.value() để lấy giá trị của biến sau khi giải, value(model.objective) để lấy giá trị hàm mục tiêu tối ưu. PuLP cũng cung cấp trạng thái lời giải (LpStatus[model.status]).

Khi code Python cho tối ưu hóa, nên chia nhỏ vấn đề và thử nghiệm từng phần: trước tiên đọc đúng dữ liệu, kiểm tra dữ liệu đầu vào; sau đó thiết lập mô hình và in thử một vài ràng buộc hoặc giá trị hàm mục tiêu để đảm bảo logic; cuối cùng mới gọi solve và phân tích kết quả. Với các mô hình phức tạp, việc debug có thể tốn thời gian, nên viết code rõ ràng, đặt tên biến dễ hiểu, và in log trong quá trình xây dựng mô hình để theo dõi.

3. Phân tích mô hình toán tối ưu hóa chi phí chọn thầu

Trước khi đi vào chi tiết cài đặt bằng Python, phần này sẽ phân tích cấu trúc **mô hình tối ưu hóa** cho bài toán chọn nhà cung cấp với chi phí và giá bậc thang. Ta xét một tình huống điển hình: có **N** nhà cung cấp tiềm năng và **M** mặt hàng (hoặc điểm nhu cầu) cần mua. Mục tiêu là thỏa mãn nhu cầu mỗi mặt hàng với chi phí tổng cộng thấp nhất, trong đó chi phí bao

gồm giá mua hàng từ nhà cung cấp *và* chi phí vận chuyển hàng về kho hoặc địa điểm cần thiết.

Các thành phần của mô hình:

- **Biến quyết định**: Ký hiệu x_{i,j} là số lượng mặt hàng *j* mua từ nhà cung cấp *i*. Đây có thể là biến liên tục (nếu chấp nhận mua phân chia không nguyên, ví dụ hàng hóa có thể mua lẻ) hoặc biến nguyên (nếu đơn vị tính bắt buộc nguyên). Trong trường hợp tổng quát, ta có thể cần biến quyết định chi tiết hơn nếu áp dụng giá bậc thang, như sẽ đề cập dưới đây.
- **Hàm mục tiêu**:
    - Nếu mỗi đơn vị hàng *j* mua từ nhà cung cấp *i* có đơn giá (bao gồm cả vận chuyển) là c_{ij} cố định, thì hàm mục tiêu là Minimize ∑_{i=1..N} ∑_{j=1..M} c_{ij} * x_{i,j} – tổng chi phí mua tất cả các mặt hàng từ các nhà cung cấp.
    - Trong trường hợp **giá bậc thang kết hợp chi phí vận tải**, đơn giá c_{ij} không còn cố định mà phụ thuộc vào lượng mua x_{i,j}. Ví dụ: 100 đơn vị đầu tiên từ nhà cung cấp *i* giá 20k/đv, 100 đơn vị tiếp theo giá 18k/đv,... hoặc chi phí vận chuyển có thể giảm nếu vận chuyển khối lượng lớn (do hiệu quả vận tải). Khi đó, hàm mục tiêu sẽ là tổng chi phí tính trên từng *phân đoạn sản lượng* tương ứng với các bậc giá. Ta phải linearize (tuyến tính hóa) hàm chi phí phân đoạn này bằng cách giới thiệu các biến phụ cho mỗi bậc giá.
- **Các ràng buộc**:

1. **Ràng buộc nhu cầu**: Mỗi mặt hàng hoặc điểm nhu cầu *j* phải được cung cấp đủ số lượng yêu cầu D_j. Do đó: ∑_{i=1..N} x_{i,j} >= D_j (thỏa mãn tối thiểu nhu cầu; nếu muốn đúng bằng nhu cầu thì là =). Mô hình hiện tại giả định phải đáp ứng ít nhất nhu cầu tối thiểu, có thể cho phép vượt nhu cầu (trong trường hợp mua dư dự trữ) hoặc không.

2. **Ràng buộc nguồn cung**: Mỗi nhà cung cấp *i* có giới hạn cung ứng tối đa S_i (do năng lực sản xuất hoặc hạn mức mà ta muốn mua). Do đó: ∑_{j=1..M} x_{i,j} <= S_i với mọi nhà cung cấp *i*. Ràng buộc này đảm bảo không đặt hàng vượt khả năng của mỗi nhà cung cấp.

3. **Ràng buộc logic cho bậc thang giá**: Đây là phần phức tạp hơn. Để mô hình hóa giá bậc thang, ta thường:

    - Chia biến x_{i,j} trên mỗi cung ứng *i,j* thành các biến thành phần tương ứng với các khoảng sản lượng. Ví dụ, giả sử nhà cung cấp *i* có các mức: 0-100 đơn vị giá A, 100-200 đơn vị giá B, >200 đơn vị giá C. Ta tạo các biến: x_{i,j,1}, x_{i,j,2}, x_{i,j,3} tương ứng lượng hàng của mặt hàng

j mua ở mức 1, 2, 3 từ nhà cung cấp *i*. Khi đó: x_{i,j} = x_{i,j,1} + x_{i,j,2} + x_{i,j,3}.

- Thêm ràng buộc giới hạn cho từng biến mức: 0 <= x_{i,j,1} <= U1*y_{i,j,1}, 0 <= x_{i,j,2} <= (U2 - L2)*y_{i,j,2},... trong đó y_{i,j,k} là biến nhị phân cho biết có sử dụng mức k hay không, U và L là giới hạn trên/dưới của khoảng. Ví dụ: nếu mức 1 là 0-100, thì U1=100. Nếu mức 2 là 100-200, thì biến mức 2 chỉ có thể dương nếu đã dùng hết mức 1 (thường mô hình cũng ép y_{i,j,1} >= y_{i,j,2} >= ... để thứ tự). Tuy nhiên, để đơn giản có thể giới hạn "at most one tier per supplier per item" (mỗi nhà cung cấp cho mỗi mặt hàng chỉ chọn một mức giá duy nhất) – tùy cách xây dựng chính sách mua.
- Các biến nhị phân y_{i,j,k} còn kèm ràng buộc: tổng y_{i,j,k} <= 1 (mỗi *i,j* chỉ chọn tối đa một bậc giá nếu áp dụng mô hình chọn một bậc), hoặc nếu cho phép cộng dồn các bậc thì phải ràng buộc thứ tự đầy đủ các bậc trước khi dùng bậc sau.

Việc mô hình hóa giá bậc thang thực sự biến bài toán thành quy hoạch tuyến tính nguyên (MILP) vì có biến nhị phân. Chiến lược phổ biến là sử dụng phương pháp Big-M để liên kết biến liên tục x và biến nhị phân y. Ví dụ, nếu khoảng [L_k, U_k] là bậc k của nhà cung cấp *i*, thì: x_{i,j,k} >= L_k * y_{i,j,k} và x_{i,j,k} <= U_k * y_{i,j,k}. Điều này đảm bảo nếu y_{i,j,k} = 0 thì mức k không được sử dụng (x=0), còn nếu y=1 thì x có thể đạt tối đa giới hạn của mức đó và tối thiểu bằng giới hạn dưới (tức là nếu đã chọn mức này thì phải mua ít nhất bằng lượng tối thiểu của mức). Đồng thời ràng buộc ∑_k y_{i,j,k} <= 1 (mỗi cặp *i,j* chọn tối đa một mức giá) để tránh mua cùng lúc ở hai bậc giá khác nhau cho cùng một nhà cung cấp – tùy chính sách. (Trong mã ví dụ ở phần sau, ta sẽ thấy các ràng buộc này được triển khai chi tiết).

- **Ràng buộc bổ sung khác**: Tùy bài toán thực tế, có thể có thêm ràng buộc như: hạn mức ngân sách tổng, ưu tiên hoặc bắt buộc mua một tỷ lệ từ nhà cung cấp địa phương, hoặc ràng buộc về số lượng nhà cung cấp được chọn tối đa,... Những ràng buộc này cũng có thể được đưa vào mô hình dưới dạng tuyến tính hoặc logic (dùng biến nhị phân).

Nhìn chung, mô hình tối ưu hóa chi phí chọn thầu kết hợp giá bậc thang và chi phí vận tải là sự kết hợp của **mô hình vận tải** (phân bổ nguồn cung cho nhu cầu với chi phí tối thiểu) (Bài toán vận tải – Wikipedia tiếng Việt) và **mô hình chiết khấu theo sản lượng**. Đây là một bài toán phức tạp hơn dạng tuyến tính chuẩn, nhưng có thể giải được bằng kỹ thuật MILP. Khi mô hình được xây dựng đúng, ta có thể sử dụng các bộ giải (solvers) để tìm phương án tối ưu một cách tự động, thay vì phải thử nghiệm thủ công nhiều kịch bản đặt hàng.

4. Thiết kế và sử dụng file Excel nhập dữ liệu động

Để thuận tiện cho việc thử nghiệm và mở rộng bài toán, chúng ta sẽ sử dụng một file Excel cho phép nhập **dữ liệu đầu vào một cách động** (có thể dễ dàng thay đổi số liệu mà không cần chỉnh sửa mã Python). Việc tách riêng dữ liệu vào file Excel giúp các nhà quản lý – những người quen dùng bảng tính – có thể cập nhật thông tin nhà cung cấp, chi phí, nhu cầu… và chạy lại mô hình để nhận kết quả mới.

Tổ chức dữ liệu trong file Excel:
Một cách cấu trúc phổ biến là sử dụng **một sheet riêng cho mỗi nhà cung cấp** để lưu bảng giá và chi phí vận chuyển theo các mức sản lượng, cùng với một số sheet tổng hợp cho nguồn cung và nhu cầu. Trong ví dụ của tài liệu này, file Excel được thiết kế như sau:

- **Sheet "Supply" (Nguồn cung)**: chứa thông tin về năng lực cung ứng tối đa của mỗi nhà cung cấp. Ví dụ:

Nhà CC	A	B	C	D	Windows
Số lượng tối đa	4000	1800	3000	3600	3000

(Dòng đầu liệt kê tên các nhà cung cấp, dòng thứ hai là khả năng cung ứng tương ứng. Ở đây có 5 nhà cung cấp ký hiệu A, B, C, D và "Windows".)

- **Sheet "Demand" (Nhu cầu)**: chứa nhu cầu cần mua đối với từng mã hàng hoặc điểm nhu cầu. Ví dụ:

Mặt hàng	B1	B2	B3	B4	B5	B6
Nhu cầu	350	2800	2000	1200	3000	2000

(Ở đây có 6 mặt hàng/điểm nhu cầu ký hiệu B1–B6 với các số lượng nhu cầu tương ứng.)

- **Sheet cho từng nhà cung cấp (ví dụ: "A", "B", "C", "D", "Windows")**: Mỗi sheet này chứa **bảng giá bậc thang và chi phí vận chuyển** của nhà cung cấp đó cho các mặt hàng. Cấu trúc bảng gồm hai phần: bên trái là các mức sản lượng và đơn giá, bên phải là chi phí vận chuyển.

Ví dụ, trích đoạn sheet **"A"** (Nhà cung cấp A):

Bảng giá theo sản lượng (nhà cung cấp A):

Từ (đv)	Đến (đv)	B1	B2	B3	B4	B5	B6
0	100	15	20	17	24	20	24
100	200	13	18	16	22	18	22
200	500	12	16	14	19	16	19
500	600	10	13	12	16	12	16
600	10000	9	11	10	15	10	15

Bảng chi phí vận chuyển (nhà cung cấp A):

Từ (đv)	Đến (đv)	B1	B2	B3	B4	B5	B6
0	100	4	8	10	5	8	10
100	200	10	3	6	8	10	5
200	500	9	7	8	3	7	4
500	600	3	3	5	6	5	4
600	10000	8	10	7	8	6	3

Giải thích cấu trúc trên: Nhà cung cấp A có các mức sản lượng 0-100, 100-200, 200-500, 500-600, 600-10000 đơn vị. Ứng với mỗi mức, đơn giá cho các mặt hàng B1…B6 thay đổi (nhìn bảng giá: mua càng nhiều thì giá đơn vị giảm dần, ví dụ mặt hàng B1 giá 15 khi mua <=100 đơn vị, giảm còn 13 khi mua 100-200 đơn vị, v.v.). Đồng thời, chi phí vận chuyển trên mỗi đơn vị hàng cũng phụ thuộc vào lượng vận chuyển (bảng chi phí vận chuyển bên phải). Chẳng hạn, vận chuyển 0-100 đơn vị mặt hàng B1 có chi phí 4 (nghìn đồng)/đv, nhưng nếu vận chuyển 100-200 đơn vị thì chi phí tăng lên 10/đv (do có thể phải dùng phương tiện khác, ví dụ không đầy tải). Đây chỉ là số liệu giả lập; thực tế chi phí vận chuyển thường tăng ít hoặc giảm khi khối lượng nhiều, nhưng ví dụ này cho thấy chi phí không nhất thiết đơn điệu mà có thể phụ thuộc nhiều yếu tố (chẳng hạn thuê xe tải nguyên chiếc rẻ hơn khi đủ tải).

Lý do tổ chức file theo cách này: Cách bố trí trên cho phép:

- Dễ dàng thêm/bớt nhà cung cấp chỉ bằng việc thêm hoặc bớt một sheet, đồng thời cập nhật sheet Supply (nguồn cung). Mã Python có thể được viết để tự động đọc tất cả các sheet nhà cung cấp hiện có.
- Mỗi nhà cung cấp có thể có cơ cấu giá riêng, số mức bậc thang tùy ý. Ví dụ trong file, nhà cung cấp B, C, D cũng có bảng cấu trúc có thể khác về khoảng sản lượng và giá.
- Việc tách riêng bảng giá và bảng chi phí vận tải (mặc dù đặt cạnh nhau trên cùng sheet) giúp rõ ràng hai phần cấu thành chi phí. Nếu trong trường hợp chỉ có giá sản phẩm (không tách chi phí vận chuyển) thì có thể bỏ qua phần chi phí vận chuyển (hoặc để bằng 0).
- **Tính mở rộng**: Người dùng có thể sửa file Excel – ví dụ điều chỉnh nhu cầu, thêm mặt hàng B7 (bằng cách thêm cột trong sheet Demand và thêm cột tương ứng trong mỗi sheet nhà cung cấp với giá và phí), sau đó chạy lại chương trình Python mà không cần sửa code (miễn là code được viết để duyệt qua các cột một cách động). Tương tự, thêm nhà cung cấp mới (thêm sheet mới và thêm cột trong Supply) cũng sẽ được mã tự động cập nhật.

Khi thiết kế file Excel động, lưu ý:

- Đặt tên sheet và tiêu đề nhất quán để mã chương trình dễ suy luận. Ở đây, sheet "Supply" hàng đầu tiên liệt kê tên nhà cung cấp chính xác khớp với tên sheet của từng nhà cung cấp (A, B, C, D, Windows), và sheet "Demand" liệt kê mã mặt hàng khớp với các cột trong bảng giá/chi phí của các sheet nhà cung cấp.
- Tránh để các ô trống hoặc dữ liệu thừa ngoài bảng, vì khi đọc bằng code có thể bị nhiễu. Trong ví dụ, cột "Unnamed: 8" là cột trống ngăn cách hai bảng trong sheet, khi đọc vào Python có thể cần loại bỏ.
- Luôn có một khoảng giá cuối cùng lớn đủ bao phủ mọi khả năng (như 600-10000 trong ví dụ) để nếu nhu cầu rất lớn vẫn có bậc giá để áp dụng, tránh trường hợp vượt ngoài định nghĩa gây sai mô hình.
- Kiểm tra kỹ dữ liệu trong Excel trước khi chạy tối ưu: tổng cung có đáp ứng tổng cầu không? (Nếu tổng cung thiếu có thể không thỏa mãn nhu cầu, mô hình có thể không khả thi hoặc cho kết quả không đủ hàng). Nếu tổng cung thừa rất nhiều so với cầu, có thể bổ sung một "nhà cung cấp giả" hoặc "nhu cầu giả" để cân bằng, nhưng trong mô hình MILP solver vẫn có thể giải ngay cả khi cung > cầu bằng cách để một số cung không dùng đến.

5. Chi tiết code Python (liên hệ trực tiếp ERX Vietnam)

6. Mở rộng bài toán trong thực tế

Để hiểu rõ hơn mô hình, hãy xem xét một vài **kịch bản thực tế** và cách điều chỉnh:

- **Kịch bản 1: Chi phí vận chuyển không phụ thuộc khối lượng** – Giả sử trong nhiều trường hợp, chi phí vận chuyển tính theo đơn vị sản phẩm không đổi (ví dụ gửi hãng vận chuyển tính theo kg tuyến cố định). Ta có thể đơn giản hóa mô hình bằng cách nhập mọi chi phí vận chuyển giống nhau cho các mức, hoặc thậm chí gộp chi phí vận chuyển vào đơn giá và bỏ hẳn phần biến y nếu giá sản phẩm cũng không có bậc thang. Mô hình sẽ trở thành một bài toán vận tải tiêu chuẩn: kết quả thường sẽ mua tối đa từ nhà cung cấp nào có chi phí (giá + vận chuyển) rẻ nhất cho từng mặt hàng cho đến khi hết công suất, sau đó chuyển sang nhà cung cấp rẻ thứ hai, v.v. (theo *quy tắc xếp hạng chi phí*).
- **Kịch bản 2: Nhu cầu vượt tổng cung** – Nếu tổng nhu cầu vượt tổng cung của tất cả nhà cung cấp (ví dụ Demand tăng cao), mô hình có thể không tìm được lời giải khả thi (infeasible). Lúc này người quản lý cần xem xét: hoặc bổ sung nhà cung cấp mới, hoặc điều chỉnh nhu cầu (mức độ chấp nhận thiếu hàng). Có thể mô hình hóa thiếu hàng bằng cách thêm biến *shortfall* (thiếu hụt) cho mỗi nhu cầu và cho phép mua không đủ, nhưng phải thêm chi phí phạt cho shortfall để mô hình vẫn tối ưu hóa (chi phí phạt rất lớn để tránh thiếu trừ khi bất khả kháng).
- **Kịch bản 3: Thay đổi số lượng nhà cung cấp, mặt hàng** – Giả sử thêm một nhà cung cấp E mới với bảng giá tương tự A, B,... Bạn chỉ cần tạo thêm sheet E trong file Excel, điền dữ liệu cung ứng vào sheet Supply, sau đó chạy lại chương trình. Code được viết tổng quát sẽ tự động thêm nhà cung cấp E vào biến suppliers và xây dựng các biến, ràng buộc tương ứng. Tương tự, nếu thêm mặt hàng mới B7, ta thêm vào sheet Demand và cột dữ liệu cho mỗi sheet nhà cung cấp.
- **Kịch bản 4: Nhà cung cấp đưa ra chiết khấu mạnh sau một ngưỡng** – Ví dụ một nhà cung cấp D cung cấp mặt hàng B5, giá 10k/đv cho đến 1000 đv, nhưng nếu mua trên 1000 đv thì *toàn bộ* đơn giá giảm còn 8k/đv (áp dụng hồi tố). Trường hợp này không phải volume discount thông thường (mà là **bậc thang hồi tố** – retroactive discount). Mô hình trên chưa trực tiếp xử lý hồi tố (vì hiện tại mỗi mức chỉ áp dụng cho phần vượt). Để mô hình hóa, có thể cần tạo biến nhị phân để chọn *phương án giá*: hoặc mua <1000 với giá 10k, hoặc mua >=1000 với giá 8k cho toàn bộ. Đây là dạng chọn lựa rời rạc khác, có thể thiết lập bằng 2 biến nhị phân đại diện cho hai kịch bản và ràng buộc logic để nếu chọn kịch bản giảm giá thì x >=1000, và chi phí

tính theo 8k * x; còn nếu không chọn thì x <=1000 và chi phí 10k * x. Điều này hơi khác mô hình gốc nhưng có thể tích hợp bằng cách mở rộng tập biến y.

- **Kịch bản 5: Đa mục tiêu hoặc ràng buộc bổ sung** – Trong thực tế, quyết định chọn nhà cung cấp còn có thể dựa trên nhiều yếu tố: thời gian giao hàng, chất lượng, rủi ro,... Mô hình trên chỉ tối ưu chi phí. Để đưa các yếu tố khác vào, ta có thể:
    - Thêm các ràng buộc về số nhà cung cấp tối đa được chọn (ví dụ hạn chế chỉ chọn 2 trong 5 nhà cung cấp để đơn giản hóa quản lý). Ràng buộc này cần biến nhị phân đại diện cho việc "nhà cung cấp s có được sử dụng hay không", rồi ∑ y_supplier <= 2. Liên kết y_supplier với x: nếu bất kỳ x[s,b,i] >0 thì y_supplier=1.
    - Thêm mục tiêu phụ như **tối ưu thời gian giao hàng** có thể bằng cách đặt trọng số chi phí phạt cho đơn hàng giao chậm, hoặc chạy mô hình nhiều lần: lần 1 tối ưu chi phí, lần 2 cố định chi phí ~ tối ưu và tối ưu hóa tiếp thời gian.
    - Sử dụng phương pháp **đa mục tiêu (multi-objective)** hoặc **phân tích kịch bản**: Chạy mô hình nhiều lần với các trọng số khác nhau giữa chi phí và các yếu tố khác để thấy đánh đổi (trade-off). Ví dụ, nếu muốn giảm rủi ro phụ thuộc một nhà cung cấp, có thể thêm vào hàm mục tiêu một chi phí giả cho việc sử dụng nhà cung cấp vượt một ngưỡng nào đó.

Sau khi có kết quả mô hình (phương án tối ưu), nhà quản lý nên **phân tích kết quả**:

- Nhà cung cấp nào được chọn và cung cấp bao nhiêu? Có bất ngờ không (ví dụ một nhà cung cấp giá rẻ nhưng không được chọn vì phí vận chuyển cao hoặc vì giới hạn công suất).
- Chi phí tổng bao nhiêu và bao gồm những khoản nào? Việc tách chi tiết kết quả như đoạn code xuất output ở trên giúp hiểu rõ chi phí đóng góp từ từng kết hợp nhà cung cấp – mặt hàng.
- **Bài học rút ra**: Có thể phát hiện ra rằng nếu tăng giới hạn cung ứng của một nhà cung cấp thêm X đơn vị sẽ giúp giảm chi phí đáng kể (vì hiện tại nhà đó dùng hết công suất và ta phải mua phần còn lại chỗ đắt hơn). Điều này là tín hiệu cho việc đàm phán nâng hạn mức hoặc tìm thêm nhà cung cấp tương tự.
- Hoặc phát hiện một mặt hàng nào đó toàn bộ mua từ một nguồn đắt, có thể cân nhắc tìm nhà cung cấp mới cho mặt hàng đó trong dài hạn.

Mô hình tối ưu cung cấp **cơ sở định lượng** để ra quyết định, nhưng người quản lý cũng cần kết hợp với thực tế (quan hệ nhà cung cấp, rủi ro, chất lượng) để đưa ra kế hoạch cuối cùng.

Tuy nhiên, việc có một công cụ tự động tính toán chi phí tối ưu giúp tiết kiệm rất nhiều thời gian so với thử nghiệm thủ công trên Excel.

7. Tài liệu tham khảo

- Winston, Wayne L. *Operations Research: Applications and Algorithms*. (Sách giáo khoa kinh điển về Quy hoạch tuyến tính và ứng dụng trong bài toán vận tải, phân bổ nguồn lực, v.v.)
- Ghodsypour, S.H. & O'Brien, C. (1998). **A decision support system for supplier selection using an integrated AHP and linear programming**. International Journal of Production Economics, 56-57, 199–212 (Model of multiple suppliers. | Download Scientific Diagram). (Bài báo khoa học về lựa chọn nhà cung cấp đa tiêu chí kết hợp phương pháp phân tích thứ bậc và lập mô hình LP)
- Ghodsypour, S.H. & O'Brien, C. (2001). **The total cost of logistics in supplier selection, under multiple sourcing, multiple criteria, and capacity constraint**. International Journal of Production Economics, 73(1), 15–27 (Model of multiple suppliers. | Download Scientific Diagram). (Bài báo nghiên cứu áp dụng mô hình quy hoạch nguyên hỗn hợp cho bài toán chọn nhà cung cấp có xét chi phí logistics và giới hạn công suất)
- Kawtummachai, R. & Van Hop, N. (2005). **Order allocation in a multiple-supplier environment**. International Journal of Production Economics, 93-94, 231–238 (Model of multiple suppliers. | Download Scientific Diagram). (Nghiên cứu về phân bổ đơn hàng trong môi trường nhiều nhà cung cấp, có xét yếu tố giảm giá khi đặt hàng phân bổ)
- Wikipedia tiếng Việt: **Bài toán vận tải** (Bài toán vận tải – Wikipedia tiếng Việt) (Bài toán vận tải – Wikipedia tiếng Việt). (Giới thiệu mô hình toán học của bài toán vận tải – trường hợp đặc biệt đơn giản của bài toán chọn nhà cung cấp khi không có bậc thang giá)
- Pandas Documentation – *pandas.read_excel* (pandas.read_excel — pandas 2.2.3 documentation - PyData |). (Hướng dẫn sử dụng hàm đọc file Excel trong Python Pandas)
- PuLP Documentation – *A Python Linear Programming API* (Optimization with PuLP - COIN-OR Documentation). (Tài liệu chính thức của PuLP, mô tả cách khai báo biến, thiết lập bài toán LP/MILP bằng Python)

- Ben Alex Keen (2016). **Linear Programming with Python and PuLP** (Linear Programming with Python and PuLP – Ben Alex Keen). *benalexkeen.com*. (Blog hướng dẫn cơ bản về lập mô hình tối ưu tuyến tính với PuLP và Python)
- Medium Article: Hobeom Lee (2021). *Transportation Optimization using Python PuLP*. (Ví dụ về bài toán vận tải giải bằng PuLP – minh họa cách sử dụng thư viện trong bài toán tối ưu chuỗi cung ứng đơn giản)
- Google Colaboratory: *Welcome to Colab*. (Tài liệu/giới thiệu Google Colab của Google, hướng dẫn cách sử dụng notebook trên mây cho Python)