# ỨNG DỤNG LẬP TRÌNH RÀNG BUỘC VÀO BÀI TOÁN XẾP LỊCH TÀI XẾ CHO CÔNG TY VẬN CHUYỂN HÀNG HÓA

**Dương Tuấn Anh[1], Nguyễn Đức Huy[2]**

[1] Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học Thành phố Hồ Chí Minh
[2] Công ty Vận chuyển hàng hóa Phan Long, Quận 11, Tp. Hồ Chí Minh
*anhdt@huflit.edu.vn, phanlongvietnam@gmail.com*

**TÓM TẮT**— Bài toán xếp lịch tài xế là một bài toán quan trọng, ảnh hưởng trực tiếp đến hiệu suất hoạt động của công ty vận chuyển hàng hóa và sự hài lòng của khách hàng. Công việc của tài xế bao gồm việc giao nhận hàng đúng thời gian, tuân thủ các quy định về giao thông và luật lệ của nhà kho. Do đó, bài toán xếp lịch tài xế không chỉ là một vấn đề phân công công việc mà còn liên quan đến việc tối ưu hóa hoạt động của toàn bộ hệ thống vận chuyển hàng hóa. Để áp dụng các kỹ thuật lập trình ràng buộc (constraint programming) vào tình huống cụ thể, là việc xếp lịch tài xế tại công ty vận chuyển Phan Long, Quận 11, Tp. Hồ Chí Minh, chúng tôi ứng dụng cách tiếp cận lập trình ràng buộc với giải thuật tối ưu hóa *nhánh-và-cận* (branch-and-bound) để giải quyết hai bài toán con quan trọng trong công tác xếp lịch này: (i) xếp lịch chuyến hàng sao cho tối ưu hóa về thời gian dựa vào *bài toán xếp lịch với các ràng buộc thứ tự trước sau* và (ii) *bài toán gán* (assignment problem) để phân công tài xế phụ trách các chuyến hàng sao cho thích hợp nhất. Kết quả thực nghiệm trên dữ liệu thực tế tại công ty Phan Long cho thấy tính ổn định, tính chính xác và tính hữu hiệu về thời gian tính toán của giải pháp đề xuất.

**Từ khóa**— *xếp lịch tài xế, vận chuyển hàng hóa, lập trình ràng buộc, giải thuật nhánh-và-cận, xếp lịch với các ràng buộc thứ tự trước sau, bài toán gán.*

## I. GIỚI THIỆU

Bài toán *xếp lịch tài xế* (driver scheduling) là một bài toán chủ yếu, có ảnh hưởng trực tiếp đến hiệu quả hoạt động của công ty vận chuyển hàng hóa và sự hài lòng của khách hàng. Công việc của tài xế bao gồm việc giao nhận hàng đúng thời gian, tuân thủ các quy định về giao thông và luật lệ của nhà kho. Do đó, bài toán xếp lịch tài xế không chỉ là một vấn đề phân công công việc mà còn liên quan đến việc tối ưu hóa hoạt động của toàn bộ hệ thống vận chuyển hàng hóa [1].

Có rất nhiều công trình nghiên cứu liên quan đến bài toán *xếp lịch tài xế* cho công ty vận chuyển hàng hóa vì bài toán này là một bài toán thành phần quan trọng trong nhiều ứng dụng có phạm vi lớn hơn, thuộc lãnh vực logistic. Một số những công trình tiêu biểu trong những năm gần đây liên quan đến bài toán xếp lịch tài xế cho công ty vận chuyển hàng hóa được tóm lược như sau.

Azadeh và các cộng sự năm 2013 [2] đề xuất giải pháp cho bài toán *xếp lịch phi hành đoàn* (flight crew scheduling) cho hãng hàng không. Xếp lịch phi hành đoàn là việc gán các thành viên của phi hành đoàn vào các chuyến bay sao cho tổng chi phí được tối thiểu hóa và tôn trọng những nội quy của ngành hàng không. Bài toán xếp lịch phi hành đoàn là một bài toán tối ưu tổ hợp có ràng buộc thuộc loại khó nên không thể giải bằng giải thuật chính xác với thời gian tính toán chấp nhận được. Azadeh và các cộng sự đã đề xuất một *giải thuật tối ưu hóa bầy đàn* (particle swarm optimization – PSO) đồng bộ hóa với một giải thuật *tìm kiếm cục bộ* (local search) để giải những bài toán toán xếp lịch phi hành đoàn qui mô lớn.

Tian và Song năm 2013 [3] đề xuất mô hình và giải thuật giải quyết bài toán *xếp lịch đội lái tàu* (crew scheduling) cho hệ thống tàu hỏa cao tốc (high-speed railway) ở Trung Quốc. Các tác giả đã chia bài toán xếp lịch đội lái tàu thành hai giai đoạn: giai đoạn thứ nhất xác định tập hợp các *lộ trình* của đội lái tàu (crew route) và giai đoạn thứ hai gán các đội lái tàu đến các lộ trình. Các tác giả đã sử dụng mô hình *phủ tập* (set-covering) và giải thuật *đàn kiến* (ant algorithm) cải tiến, một giải thuật metaheuristic, để giải quyết bài toán nêu trên. Các tác giả đã thử nghiệm mô hình đề xuất vào hệ thống tàu hỏa cao tốc Bắc Kinh-Thiên Tân.

Nghiên cứu của Han và Li năm 2014 [4], đã đề xuất phương pháp dựa vào *lập trình ràng buộc* (constraint programming) để giải bài toán *xếp lịch đội tài xế* (crew scheduling) cho các chuyến tàu trong một hệ thống giao thông công cộng tại Đài Bắc (Đài Loan). Bài toán này có mục đích tìm ra số *nhiệm vụ* (duty) nhỏ nhất bao phủ tất cả các chuyến di chuyển (task) mà thỏa mãn tất cả các ràng buộc cứng và ràng buộc mềm của việc xếp lịch. Các tác giả đề xuất một cách tiếp cận bao gồm một mô hình lập trình ràng buộc để sinh ra các nhiệm vụ, một mô hình bài toán *phủ tập* (set-covering) để tối ưu hóa số nhiệm vụ và các phương pháp khác để tìm ra lời giải cuối cùng tại những tình huống khác nhau. Các tác giả đã áp dụng cách tiếp cận đề xuất vào bài toán *xếp lịch đội tài xế* cho các chuyến tàu trong một hệ thống giao thông công cộng tại Đài Bắc.

Nghiên cứu của Jacyna và Izdebski năm 2014 [5], đã đề xuất sử dụng giải thuật *đàn kiến* (ant algorithm) để giải quyết bài toán gán tài nguyên (xe và tài xế) cho các chuyến vận chuyển hàng. Các tác giả xác lập mô hình toán cho

bài toán và mô tả các bước khi áp dụng giải thuật đàn kiến để giải bài toán gán tài xế và xe cho các chuyến vận chuyển hàng.

Nghiên cứu của Izdebski và Jacyna năm 2014 [6], đã đề xuất sử dụng *giải thuật di truyền* (genetic algorithm) để giải quyết bài toán gán tài nguyên (xe và tài xế) cho các chuyến vận chuyển hàng. Các tác giả đã định nghĩa mô hình toán cho bài toán nêu trên và mô tả các bước khi áp dụng giải thuật di truyền để giải bài toán gán tài xế và xe tải cho các chuyến vận chuyển hàng.

Jacyna và các cộng sự năm 2018 [7] nghiên cứu bài toán *gán xe cho các chuyến vận chuyển hàng* (task assignment of vehicles) của một công ty sản xuất. Bài toán này bao gồm hai giai đoạn: giai đoạn thứ nhất là xác định các chuyến vận chuyển hàng và giai đoạn thứ hai là gán xe tải cho các chuyến vận chuyển hàng. Mỗi chuyến vận chuyển hàng được định nghĩa như là một công tác vận chuyển hàng từ nhà cung cấp đến nhà kho của công ty và vận chuyển từ nhà kho của công ty đến xưởng sản xuất của công ty. Các tác giả đã định nghĩa mô hình toán cho bài toán nêu trên như là một bài toán *tối ưu hóa đa mục tiêu* (multi-objective optimization) và đề xuất áp dụng giải thuật *di truyền* (genetic algorithm) để giải quyết bài toán này.

Nghiên cứu của Cruz và các cộng sự năm 2020 [8], đề xuất phương pháp giải quyết bài toán *phân bổ xe* (vehicle allocation) cho công ty vận chuyển hàng hóa (freight transportation). Bài toán phân bổ xe bao gồm việc phân bổ xe tải để thực hiện một yêu cầu vận chuyển hàng giữa các địa điểm. Mục đích của bài toán là cực đại hóa lợi nhuận đem lại khi công việc hoàn tất. Các tác giả đề xuất sử dụng giải thuật *branch-and-price*, một cải tiến của giải thuật *nhánh và cận* (branch-and-bound), để giải quyết bài toán phân bổ xe cho công ty vận chuyển hàng hóa. Các tác giả đã thực nghiệm phương pháp đề xuất trên 30 bộ dữ liệu mẫu và kết quả cho thấy phương pháp đề xuất làm việc khá hiệu quả.

Nghiên cứu của Benli và các cộng sự năm 2022 [9], tập trung phân tích hiệu ứng của việc lựa chọn *khung thời gian* (time window) đối với bài toán phân bổ xe cho công ty vận chuyển hàng hóa. Khung thời gian ám chỉ thời khoảng với những cận trên (upper bound) và cận dưới (lower bound) cho một hoạt động diễn ra. Khung thời gian được sử dụng như một loại *ràng buộc* (constraint) thường diễn ra trong bài toán định tuyến xe cộ và bài toán vận chuyển. Các tác giả mô hình hóa bài toán phân bổ xe như là một bài toán *quy hoạch nguyên hỗn hợp* (mixed integer linear programming).

Nhìn chung, các công trình nêu trên đều phải giải quyết những bài toán tối ưu hóa khá phức tạp và qui mô dữ liệu của bài toán khá lớn nên phải sử dụng những giải thuật như quy hoạch nguyên, tối ưu hóa đa mục tiêu hoặc các giải thuật meta-heuristic như giải thuật di truyền, giải thuật đàn kiến, giải thuật tối ưu hóa bầy đàn PSO, giải thuật tìm kiếm cục bộ, v.v… Những giải thuật metaheuristic thường đem lại kết quả xấp xỉ tối ưu chứ không đem lại kết quả tối ưu chính xác. Chỉ riêng công trình [8], Cruz và các cộng sự sử dụng một biến thể cải tiến của giải thuật *nhánh-và-cận*, là giải thuật có thể đem lại kết quả tối ưu chính xác, vì bài toán trong công trình này không quá phức tạp và có quy mô dữ liệu vừa phải.

Đối với việc xếp lịch tài xế tại công ty vận chuyển Phan Long, Quận 11, Tp. Hồ Chí Minh, là bài toán có quy mô nhỏ, chúng tôi ứng dụng kỹ thuật tối ưu hóa *nhánh-và-cận* (branch-and-bound) để giải quyết hai bài toán con (subproblem) quan trọng trong công tác xếp lịch này: (i) xếp lịch chuyến hàng sao cho tối ưu hóa về thời gian dựa vào *bài toán xếp lịch với các ràng buộc thứ tự trước sau* và (ii) *bài toán gán* (assignment problem) để phân công tài xế phụ trách các chuyến hàng sao cho thích hợp nhất. Hai bài toán con nêu trên có dạng thức của bài toán thỏa hệ ràng buộc tối ưu hóa (optimized constraint satisfaction problem) và được giải quyết theo hướng tiếp cận *lập trình ràng buộc* (constraint programming). Kết quả thực nghiệm trên dữ liệu thực tế tại công ty Phan Long cho thấy tính ổn định, tính chính xác và tính hữu hiệu về thời gian tính toán của giải pháp đề xuất.

Phần tiếp theo của bài báo được tổ chức như sau: Mục II giới thiệu về các bài toán tối ưu hóa và giải thuật liên quan; mục III mô tả sự hiện thực và thực nghiệm trên dữ liệu cụ thể của công ty Phan Long; mục IV nêu một vài kết luận và hướng phát triển của nghiên cứu này.

## II. CÁC BÀI TOÁN TỐI ƯU HÓA VÀ GIẢI THUẬT LIÊN QUAN

### *A. BÀI TOÁN THỎA HỆ RÀNG BUỘC*

Trong *bài toán thỏa hệ ràng buộc* (constraint satisfaction problem -CSP), chúng ta được cho:

- Một tập biến, các *miền trị* (domain) cho các biến trong tập biến,
- Một tập các ràng buộc.

Mỗi ràng buộc được định nghĩa trên một tập con của tập biến gốc được cho và nhằm giới hạn các tổ hợp trị được phép gán cho tập con các biến có liên quan với ràng buộc.

Mục đích là tìm ra một phép *gán trị* (assignment) vào các biến sao cho thỏa tất cả các ràng buộc được cho [10]. Các bài toán CSP có thể được chia làm hai nhóm:

- Bài toán *thỏa ràng buộc* (Satisfiability problem) có mục đích tìm một phép *gán trị* (value asignment) sao cho thỏa tập ràng buộc. Một phép gán trị vào biến có thể thỏa các ràng buộc hay không.
- Bài toán *tối ưu hóa* (Optimization problem) là bài toán mà trong đó mỗi sự gán trị vào biến có một giá trị hàm chi phí hoặc hàm mục tiêu đi kèm với nó. Mục đích của bài toán là tìm một phép gán trị vào biến đạt giá trị hàm chi phí cực tiểu hoặc giá trị hàm mục tiêu cực đại. Phép gán trị đó được gọi là *phép gán tối ưu* (optimal assignment).

Nhiều bài toán trong lãnh vực trí tuệ nhân tạo, khoa học máy tính và các lãnh vực khác của khoa học kỹ thuật có thể được mô tả như là những bài toán thỏa hệ ràng buộc.

## 1. THIẾT LẬP MỘT BÀI TOÁN THỎA HỆ RÀNG BUỘC

Một bài toán thỏa hệ ràng buộc được mô tả bởi một tập biến $V_1$, $V_2$, …, $V_n$. Mỗi biến $V_i$ có một miền trị tương ứng $Dv_i$ bao gồm những trị khả hữu của biến đó.

Đối với bài toán thỏa ràng buộc, sẽ có những quan hệ ràng buộc giữa những tập con khác nhau của tập biến mà đem lại những tổ hợp trị hợp lệ để gán vào các biến. Những ràng buộc này có thể đặc tả như là những tập con của tích Đề Các của các miền trị của các biến liên quan.

Một *lời giải* (solution) của bài toán thỏa hệ ràng buộc là một bộ gồm *n* trị gán vào các biến sao cho thỏa mãn tất cả các ràng buộc của tập ràng buộc.

Đối với bài toán tối ưu hóa, có tồn tại một hàm để đánh giá chi phí cho mỗi phép gán trị vào biến. Một *lời giải* của bài toán tối ưu hóa là một bộ gồm *n* trị gán vào các biến sao cho tối ưu hóa được *hàm chi phí* (cost function).

## 2. GIẢI THUẬT QUAY LUI CHUẨN ĐỂ GIẢI BÀI TOÁN THỎA HỆ RÀNG BUỘC

Gọi *D* là tích Đề Các của các miền trị trong bài toán thỏa hệ ràng buộc:

$$D = Dv_1 \times Dv_2 \times \ldots \times Dv_n$$

Bằng *giải thuật quay lui* (backtracking algorithm), chúng ta có thể thăm dò một cách có hệ thống bằng cách gán trị vào các biến với các biến được sắp theo một thứ tự nào đó và đánh giá mỗi ràng buộc ngay khi các biến liên quan đến ràng buộc đã được gán trị. Nếu có bất kỳ ràng buộc này không được thỏa bởi sự gán trị thì *sự gán trị riêng phần* (partial assignment) vừa thực hiện không thể là một phần của sự gán trị đầy đủ và hợp lệ. Chúng ta loại bỏ sự gán trị riêng phần đó, và tiếp theo, chúng ta có thể thử áp dụng những sự gán trị riêng phần khác. Mỗi một sự gán trị riêng phần thất bại sẽ giúp thu hẹp không gian tìm kiếm rất lớn của tập hợp *D*. Giải thuật quay lui là một quá trình lặp được thực hiện cho đến khi mọi biến đều đã được gán trị một cách thành công.

### *B. GIẢI THUẬT NHÁNH-VÀ-CẬN*

Giải thuật *nhánh-và-cận* (branch-and-bound - BB) vận hành tương tự như giải thuật quay lui chuẩn. Giải thuật nhánh-và-cận theo dõi *lời giải tốt nhất tìm thấy hiện giờ* (best-so-far solution) và từ bỏ một nhánh tìm kiếm khi phát hiện nhánh này không thể dẫn đến một lời giải tốt hơn lời giải tốt nhất tìm thấy hiện giờ.

Giải thuật nhánh-và-cận dựa trên một phiên bản đặc biệt của giải thuật quay lui chuẩn mà *tìm kiếm vét cạn* mọi lời giải của bài toán chứ không phải chỉ lời giải đầu tiên. Dựa trên phiên bản tìm kiếm vét cạn của giải thuật quay lui chuẩn, giải thuật nhánh-và-cận có thể tìm thấy lời giải tốt nhất cho bài toán CSP tối ưu hóa.

Giải thuật nhánh-và-cận để giải bài toán CSP tối ưu hóa sử dụng một *hàm đánh giá* (thường là hàm chi phí) để đánh giá độ tốt của lời giải tìm thấy.

Cũng giống như giải thuật quay lui chuẩn, giải thuật nhánh-và-cận có độ phức tạp tính toán khá cao (hàm mũ) nên giải thuật này chỉ thích hợp với những bài toán tối ưu hóa có quy mô nhỏ hoặc vừa phải và không thích hợp với những bài toán tối ưu hóa phức tạp và có quy mô lớn.

### *C. BÀI TOÁN XẾP LỊCH VỚI CÁC RÀNG BUỘC THỨ TỰ TRƯỚC SAU*

Bài toán *xếp lịch với các ràng buộc thứ tự trước sau* (scheduling problem with precedence constraints) là một bài toán thường gặp trong nhiều ứng dụng thực tế [11], được định nghĩa như sau. Chúng ta được cho một dự án gồm:

- Một tập gồm *n* công tác (task) mà công tác *i* có thời lượng là $d_i$ và
- Một tập các ràng *buộc thứ tự trước sau* (precedent constraint) giữa các cặp công tác.

Một ràng buộc thứ tự trước sau giữa hai công tác *i* and *j* hàm ý rằng công tác *j* được phép bắt đầu sau khi công tác *i* hoàn tất.

Dự án bắt đầu từ thời điểm 0, và bài toán xếp lịch phải tìm kiếm một lịch biểu làm việc sao cho cực tiểu hóa tổng thời lượng của toàn dự án.

Chúng ta thêm vào một *công tác giả* (fictitious task) với thời lượng 0, được gọi là *công tác kết thúc* (end task) mà được đi trước bởi mọi công tác khác.

Thời lượng của các công tác có thể được dùng để tính ra hai đại lượng quan trọng của mỗi công tác: *thời điểm bắt đầu sớm nhất* (earliest time) và *thời điểm bắt đầu muộn nhất* (latest time)

Thời điểm bắt đầu sớm nhất là thời điểm mà tại đó công tác bắt đầu nếu những công tác đi trước công tác này đã được bắt đầu sớm nhất có thể. Thời điểm bắt đầu sớm nhất $t_i$ cho công tác $i$ là:

$$t_i = \max (t_j + d_j)$$

với mọi công tác $j$ đi trước công tác $i$.

Suy ra $t_{end}$ là thời điểm bắt đầu của công tác kết thúc (end task) và cũng là thời lượng tối thiểu của toàn dự án.

Nếu thời lượng của toàn dự án là $t_{end}$, thời điểm bắt đầu muộn nhất $T_i$ cho công tác $i$ được cho bởi:

$$T_i = \min (T_j - d_i)$$

với mọi công tác $j$ đi sau công tác $i$, giả định rằng $T_{end}$ thì bằng với $t_{end}$.

Như vậy thời điểm bắt đầu công tác $i$ có thể được biểu diễn bằng tầm trị $[t_i, T_i]$.

*Độ lơi* (slack) $m_i$ của công tác $i$ được định nghĩa bằng độ sai biệt giữa thời điểm bắt đầu sớm nhất và thời điểm bắt đầu muộn nhất của công tác. Những công tác có độ lơi 0 được gọi là những *công tác tới hạn* (critical tasks).

Cách biểu diễn bài toán xếp lịch nêu trên được thực hiện như sau. Với mỗi công tác $i$, ta đưa vào biến $S_i$ biểu diễn thời điểm bắt đầu công tác $i$. Và ràng buộc thứ tự trước sau giữa công tác $i$ và công tác $j$ được biểu diễn như sau:

$$S_j >= S_i + d_i$$

Ví dụ ở Bảng 1 mô tả một dự án phần mềm bao gồm việc lập trình 6 module A, B, C, D, E, F.

Bảng 1. *Danh sách các module A, B, C, D, E, F trong một dự án phần mềm và quan hệ thứ tự trước sau*

| Tên module | Thời lượng (tuần) | Những module đi trước |
|---|---|---|
| A | 7 | - |
| B | 3 | A |
| C | 4 | B |
| D | 8 | A |
| E | 5 | C, D |
| F | 4 | C, D, E |

Bài toán xếp lịch thực hiện các module để hoàn tất dự án nêu trên được định nghĩa thành bài toán thỏa hệ ràng buộc như sau:

Tập biến L gồm 7 biến: SA: thời điểm bắt đầu module A, SB: thời điểm bắt đầu module B, SC: thời điểm bắt đầu module C, SD: thời điểm bắt đầu module D, SE: thời điểm bắt đầu module E, SF: thời điểm bắt đầu module F, Send: thời điểm kết thúc toàn dự án,

Miền trị của mỗi biến là 0..30, nghĩa là thời điểm bắt đầu của mọi module không được phép vượt quá tuần thứ 30.

Tập các ràng buộc như sau:

SB >= SA+7, SC >= SB+3, SD >= SA+7,

SE >= SC+4, SE >= SD+8,

SF >= SC+4, SF >= SD+8, SF >= SE+5, Send >= SF+4.

Mục tiêu của bài toán trên là gán các giá trị thời gian vào các biến SA, SB, SC, SD, SE, SF, Send sao cho thỏa mãn tập ràng buộc nêu trên và đồng thời *Send* đạt giá trị nhỏ nhất.

Để tìm ra lời giải tối ưu chính xác cho bài toán xếp lịch này (là một bài toán thỏa hệ ràng buộc -CSP), chúng ta có thể sử dụng giải thuật *nhánh-và-cận* được mô tả ở mục II.B.

### *D. BÀI TOÁN GÁN*

*Bài toán gán* (assignment problem) là bài toán gán *n con người* (person) vào *n công việc* (job) sao cho tổng chi phí là nhỏ nhất [12]. Để thích hợp với bài toán gán, các ứng dụng cần phải được thiết lập sao cho thỏa mãn các giả định sau đây:

- Số người và số công việc phải bằng nhau (con số này được ký hiệu là *n*.)
- Mỗi con người chỉ được gán đến một công việc.
- Mỗi một công việc chỉ được thực hiện bởi một con người.
- Có chi phí $c_{ij}$ khi con người *i* (*i* = 1,2, …, n) thực hiện công việc *j* (*j* =1,2, …, n).
- Mục tiêu của bài toán là xác định *n* sự gán sao cho tối thiểu hóa hàm chi phí.

Ví dụ: Một công ty cần phân công 4 nhân viên vào 4 công việc. Các chi phí $c_{ij}$ khi nhân viên *i* (*i* = 1,2, …,4) thực hiện công việc *j* (*j* =1,2, …,4) được cho như Bảng 2.

Bảng 2 *Ví dụ về bài toán gán*

| | | Công việc | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| Nhân viên | 1 | 13 | 16 | 12 | 11 |
| | 2 | 15 | M | 20 | 13 |
| | 3 | 5 | 7 | 10 | 6 |
| | 4 | 7 | 6 | 9 | 4 |

Lưu ý: Một chi phí rất lớn *M* được gắn với sự gán nhân viên thứ 2 vào công việc thứ hai nếu như sự gán này thuộc diện bị cấm.

Bài toán gán của thí dụ nêu trên được định nghĩa thành bài toán thỏa hệ ràng buộc như sau:

Bài toán cần 16 *biến quyết định* (decision variable) (là biến chỉ có thể lấy giá trị là 0 hay là 1) với *Xij* lấy giá trị 1 nếu nhân viên *i* được gán đến công việc *j* và lấy giá trị 0 nếu ngược lại nhân viên *i* không được gán đến công việc *j*. Danh sách 16 biến quyết định như sau: L= [X11, X12, X13, X14, X21, X22, X23, X24, X31, X32, X33, X34, X41, X42, X43, X44]. Và miền trị của mỗi biến quyết định là {0,1}

Tập các ràng buộc gồm 8 ràng buộc như sau:

X11+X12+X13+X14 =1, X21+X22+X23+X24 =1, X31+X32+X33+X34 =1, X41+X42+X43+X44 =1,

X11+X21+X31+X41 =1, X12+X22+X32+X42 =1, X13+X23+X33+X43 =1, X14+X24+X34+X44 =1

Tổng chi phí của bài toán gán nêu trên là 13*X11+16*X12+12*X13+11*X14 + 15*X21 +1000*X22 + 13*X23 + 20*X24 + 5*X31 + 7*X32 + 10*X33 + 6*X34 + 7*X41 + 6*X42 + 9*X43 + 4*X44.

Mục tiêu của bài toán trên là gán các giá trị 0 hoặc 1 vào 16 biến quyết định sao cho thỏa mãn 8 ràng buộc nêu trên và đồng thời làm cho *hàm tổng chi phí* đạt giá trị nhỏ nhất. Lưu ý giá trị chi phí rất lớn M trong ma trận chi phí được thể hiện bằng giá trị 1000. Kết quả của bài toán gán thí dụ như sau: X14 = 1, X23 = 1, x31 = 1 và X42 = 1 trong khi 12 biến quyết định khác đều bằng 0. Tức là nhân viên thứ 1 được gán vào công việc 4, nhân viên thứ 2 được gán vào công việc 3, nhân viên thứ 3 được gán vào công việc 1 và nhân viên thứ 4 được gán vào công việc 2.

Để tìm ra lời giải tối ưu chính xác cho bài toán gán (là một bài toán thỏa hệ ràng buộc tối ưu hóa), chúng ta có thể sử dụng giải thuật *nhánh-và-cận* đã được mô tả ở tiểu mục II.B.

### *E. LẬP TRÌNH RÀNG BUỘC VÀ NGÔN NGỮ LẬP TRÌNH LOGIC CÓ RÀNG BUỘC*

*Lập trình ràng buộc* (constraint programming) thuộc về thể loại *lập trình khai báo* (declarative programming).

Một *ràng buộc* (constraint) biểu diễn mối liên hệ giữa các đối tượng khác nhau được định nghĩa trong một *miền tính toán* (computation domain).

Việc đưa khái niệm ràng buộc vào lập trình đem lại hai tiện lợi sau đây: i) cho phép mô tả bài toán một cách tự nhiên hơn, và ii) cho phép vận dụng những giải thuật chuyên dụng đã có ở những lãnh vực khác nhau như: trí tuệ nhân tạo, vận trù học (operations research) và toán học.

*Lập trình ràng buộc* (constraint programming) là lãnh vực nghiên cứu về những hệ thống dựa vào ràng buộc. Ý tưởng chính của lập trình ràng buộc là phát biểu các ràng buộc và sau đó tìm kiếm lời giải thỏa mãn tất cả mọi ràng buộc.

Ngôn ngữ lập trình logic (Logic Programming - LP), như là ngôn ngữ Prolog, với cách biểu diễn *mệnh đề Horn* tổng quát (general purpose Horn clause) và một ngữ nghĩa rõ ràng, có thể biểu diễn tốt những ràng buộc dưới dạng thức khai báo. Tuy vậy, *cơ chế hợp nhất* (unification mechanism) của Prolog không đủ mạnh để xử lý các ràng buộc một cách đúng đắn. Phép hợp nhất không thể xử lý ràng buộc một cách *không-tất-định* (non-deterministically).

Đã có nhiều nỗ lực trong cộng đồng nghiên cứu về lập trình logic nhằm mở rộng Prolog để khắc phục những hạn chế của phép hợp nhất. Tuy nhiên, với những cải tiến như vậy, ngôn ngữ lập trình logic vẫn không thể giải quyết các ràng buộc một cách hữu hiệu.

Sau này, có một cách tiếp cận mang tính chất đột phá mở rộng mô thức lập trình logic thành *ngôn ngữ lập trình logic có ràng buộc* (Constraint Logic Programming language). Ngôn ngữ lập trình logic có ràng buộc (viết tắt là CLP) là sự kết hợp hai mô thức lập trình khai báo: *lập trình ràng buộc* (constraint programming) và *lập trình logic* (logic programming) ([13], [14]).

Giống như lập trình logic, CLP vẫn dùng *nguyên lý phân giải* (resolution principle). Tuy nhiên, phép toán hợp nhất (unification) được thay thế bằng kỹ thuật thỏa hệ ràng buộc trên một miền ứng dụng cụ thể nào đó.

CLP cải tiến lập trình logic khi giải quyết *bài toán tìm kiếm có ràng buộc* (constrained search problem), bằng cách cung cấp: i) Những *ngôn ngữ tốt về diễn đạt* trong đó người dùng có thể làm việc trực tiếp với các đối tượng từ một miền ứng dụng nào đó và dùng những ràng buộc tự nhiên cho miền ứng dụng này, và ii) Một sự thực thi hữu hiệu hơn bằng những phương pháp và giải thuật chuyên dụng để giải hệ ràng buộc.

CHIP, CLP(R), B-Prolog, SWI-Prolog, ECL$^i$PS$^e$ v.v. là những ngôn ngữ lập trình logic có ràng buộc tiêu biểu.

## III. HIỆN THỰC VÀ THỰC NGHIỆM TRÊN DỮ LIỆU CỦA CÔNG TY PHAN LONG

Tình huống được khảo sát trong nghiên cứu này là từ Công ty vận chuyển hàng hóa Phan Long, Quận 11, Tp. Hồ Chí Minh.

### *A. MÔ TẢ CHUYẾN HÀNG TẠI CÔNG TY VẬN CHUYỂN PHAN LONG*

Tại Công ty vận chuyển hàng hóa Phan Long có 3 loại chuyến hàng (transport task) bao gồm: chuyến hàng xuất, chuyến hàng nhập và chuyến hàng kết hợp.

#### 1. CHUYẾN HÀNG XUẤT

Để thực hiện một chuyến hàng xuất tại Công ty vận chuyển hàng hóa Phan Long cho một khách hàng nào đó, người tài xế được phân công phụ trách chuyến hàng sẽ phải thực hiện các *công đoạn* (subtask) sau đây.

S1. Tài xế xuất phát từ bãi đỗ xe đến địa điểm cung cấp container để tiếp nhận container rỗng (địa điểm này thường là cảng hoặc Inland Container Depot (ICD)).

S2. Tại điểm cung cấp container, người của địa điểm cung cấp container thực hiện công việc gắp container rỗng lên phương tiện cho tài xế, tốn một thời gian nhất định.

S3. Từ địa điểm cung cấp container tài xế di chuyển xe đến nhà kho của khách hàng.

S4. Tại kho, người của công ty khách hàng đảm nhiệm công việc *lên hàng* (loading), tốn một thời gian nhất định.

S5. Từ điểm kho, tài xế di chuyển đến địa điểm trả container để bàn giao lại container có hàng (địa điểm này thường là cảng hoặc ICD).

S6. Tại điểm trả container, người của địa điểm trả container thực hiện công việc gắp container có hàng xuống rời khỏi phương tiện cho tài xế, tốn một thời gian nhất định.

S7. Từ điểm trả container, tài xế di chuyển về bãi đậu xe.

#### 2. CHUYẾN HÀNG NHẬP

Để thực hiện một chuyến hàng nhập tại Công ty vận chuyển hàng hóa Phan Long cho một khách hàng nào đó, người tài xế được phân công phụ trách chuyến hàng sẽ phải thực hiện các công đoạn sau đây.

S1. Tài xế xuất phát từ bãi đỗ xe đến địa điểm cung cấp container để tiếp nhận container có hàng (địa điểm này thường là cảng hoặc ICD).

S2. Tại điểm cung cấp container, người của địa điểm cung cấp container thực hiện công việc gắp container có hàng lên phương tiện cho tài xế, tốn một thời gian nhất định.

S3. Từ địa điểm cung cấp container tài xế di chuyển đến kho của khách hàng.

S4. Tại kho, người của công ty khách hàng đảm nhiệm công việc *xuống hàng* (unloading) trong container, tốn một thời gian nhất định.

S5. Từ điểm kho, tài xế di chuyển đến địa điểm trả container để bàn giao lại container rỗng (địa điểm này thường là cảng hoặc ICD).

S6. Tại điểm trả container, người của địa điểm trả container thực hiện công việc gắp container rỗng xuống phương tiện cho tài xế, tốn một thời gian nhất định.

S7. Từ điểm trả container, tài xế di chuyển về bãi đậu xe.

**3. CHUYẾN HÀNG KẾT HỢP**

Để thực hiện một chuyến hàng kết hợp tại công ty vận chuyển hàng hóa Phan Long, người tài xế được phân công phụ trách chuyến hàng sẽ phải thực hiện các công đoạn như sau.

S1. Tài xế thực hiện từ công đoạn S1 đến S6 của chuyến hàng trước.

S2. Tài xế di chuyển từ địa điểm ở công đoạn S6 của chuyến hàng trước đến địa điểm ở công đoạn S2 của chuyến chuyển hàng sau.

S3. Tài xế thực hiện từ công đoạn S2 đến S7 của chuyến hàng sau.

Ưu điểm khi có chuyến hàng kết hợp là tiết kiệm được chi phí, thời gian di chuyển từ điểm trả container về bãi đậu xe và từ bãi đậu xe đến điểm cung cấp container.

***B. NGÔN NGỮ LẬP TRÌNH VÀ CÁC CÔNG CỤ PHẦN MỀM HỖ TRỢ***

**1. ASP.NET**

ASP.NET là một framework phát triển ứng dụng web được phát triển bởi Microsoft. Nó cung cấp một môi trường phát triển mạnh mẽ để xây dựng các ứng dụng web động, linh hoạt và hiệu quả. ASP.NET là một phần của nền tảng .NET của Microsoft, và được sử dụng rộng rãi trên toàn thế giới cho việc phát triển các ứng dụng web từ nhỏ đến lớn [15].

Một điểm nổi bật của ASP.NET là ASP.NET hỗ trợ sử dụng nhiều ngôn ngữ lập trình như Visual Basic.NET, C#, F#.

**2. LINQ**

LINQ (Language Integrated Query) là một phần của .NET Framework của Microsoft, giúp các lập trình viên thực hiện các truy vấn dữ liệu trong các ngôn ngữ lập trình C#, Visual Basic.NET một cách dễ dàng và linh hoạt. LINQ giúp thực hiện các truy vấn dữ liệu bằng cách sử dụng cú pháp tương tự như SQL (ngôn ngữ truy vấn trong hệ cơ sở dữ liệu), nhưng được viết trực tiếp vào ngôn ngữ lập trình, giúp các đoạn mã trở nên dễ đọc hơn.

Một số điểm nổi bật của LINQ:

- LINQ được tích hợp trực tiếp vào C# và Visual Basic.NET, giúp các lập trình viên thực hiện truy vấn trực tiếp trên mã nguồn mà không cần sử dụng các ngôn ngữ truy vấn riêng biệt như SQL.
- Hỗ trợ nhiều tác vụ xử lý dữ liệu: LINQ cung cấp một số phương thức mở rộng (extension methods) và toán tử để thực hiện các thao tác trên dữ liệu như lọc, sắp xếp, nhóm, và tính toán tổng, giúp việc xử lý dữ liệu trở nên đơn giản và tiện lợi.
- Hỗ trợ nhiều nguồn dữ liệu: LINQ có thể được sử dụng để thực hiện các truy vấn trên nhiều loại nguồn dữ liệu như mảng, danh sách, bảng dữ liệu XML, và cơ sở dữ liệu.

**3. SWI-PROLOG**

SWI-Prolog là ngôn ngữ lập trình logic có ràng buộc ([16], [17], [18]), được thiết kế và triển khai các ứng dụng có tính chất trí tuệ nhân tạo và dựa vào logic.

Một số điểm nổi bật của SWI-Prolog:

- Mã nguồn mở: SWI-Prolog là một dự án mã nguồn mở, miễn phí và được phát triển liên tục bởi cộng đồng. Điều này có nghĩa là bất kỳ ai cũng có thể sử dụng và đóng góp vào việc phát triển hệ thống này.
- Đa nền tảng: SWI-Prolog có thể làm việc trên nhiều nền tảng như Windows, macOS và Linux.
- Thư viện mở rộng: SWI-Prolog đi kèm với một loạt các thư viện mở rộng cho nhiều mục đích khác nhau như xử lý ngôn ngữ tự nhiên, giao tiếp mạng, và phát triển ứng dụng web.

Thư viện CLPFD (Constraint Logic Programming over Finite Domains) của SWI-Prolog cung cấp cách tiếp cận mạnh mẽ để giải quyết các bài toán *thỏa hệ ràng buộc trên miền trị hữu hạn*. Điều này cho phép xác định giá trị các biến và ràng buộc giữa chúng bằng cách sử dụng các quy tắc logic.

Thư viện CLPFD cung cấp công cụ như các ràng buộc số nguyên, ràng buộc số nguyên không âm, ràng buộc phân biệt và nhiều loại ràng buộc khác để giải quyết bài toán thỏa hệ ràng buộc trong miền trị hữu hạn (finite domain) một cách hiệu quả.

Hai hàm thư viện của SWI-Prolog thường hay được sử dụng trong nghiên cứu này là hai hàm sau đây:

Vị từ **Labeling** với cú pháp sử dụng: labeling (Option, Vars).

Trong SWI-Prolog, vị từ *labeling* rất quan trọng trong các bài toán tìm kiếm, đặc biệt là trong việc giải quyết các bài toán thỏa hệ ràng buộc. Vị từ *labeling* được sử dụng để gán trị cho các biến *Vars* trong các tập lời giải mà hệ thống SWI-Prolog đang tìm thấy. Vị từ *labeling* được sử dụng để gán giá trị cụ thể cho các biến này dựa trên các ràng buộc đã được thiết lập. Việc sử dụng *labeling* là rất quan trọng để thu được kết quả cuối cùng của các lời giải. Nó giúp xác định các giá trị cụ thể của các biến mà thỏa mãn tất cả các ràng buộc đã được đặt ra.

Vị từ **Min** với cú pháp sử dụng: min (List, MinValue)

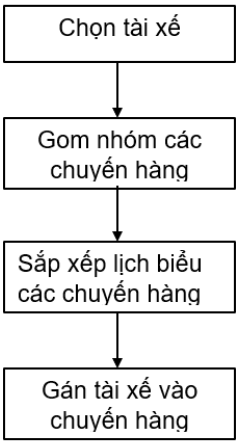
Trong SWI-Prolog, vị từ *min* được sử dụng để tìm các giá trị phù hợp cho các biến trong danh sách biến *List* sao cho biến *MinValue* đạt giá trị cực tiểu. Điều này giúp cho *MinValue* đạt giá trị nhỏ nhất ứng với các biến trong danh sách. Vị từ *min* thường được sử dụng trong các tình huống tìm giá trị nhỏ nhất của một tập hợp các giá trị có thể có trong ngôn ngữ SWI-Prolog. Lưu ý: vị từ *min* của SWI-Prolog sử dụng giải thuật *nhánh-và-cận* để giải quyết mục tiêu tối ưu hóa của vị từ này.

#### 4. GOONG DISTANCE MATRIX API

Goong Distance Matrix API là một dịch vụ được cung cấp bởi GOONG, cho phép người dùng tính khoảng cách và thời gian đi lại giữa 2 hoặc nhiều địa điểm. Dịch vụ này cung cấp thông tin về các tùy chọn giao thông như ước tính thời gian đi lại bằng ô tô, xe đạp, xe taxi, xe tải. Các tính năng này tương tự như các tính năng Google Maps Distance Matrix API.

### *C. CÁC BƯỚC THỰC HIỆN CỦA BÀI TOÁN XẾP LỊCH TÀI XẾ*

Bài toán xếp lịch tài xế cho công ty vận chuyển hàng hóa gồm bốn bước, được mô tả như trong Hình 1 sau đây:




Hình 1. *Các bước trong tiến trình xếp lịch tài xế cho công ty vận chuyển.*

#### 1. CHỌN TÀI XẾ THỰC HIỆN CÔNG VIỆC

Người điều hành cần chọn những tài xế đã hoàn thành công việc trước đó để tiếp tục thực hiện phân công xếp lịch chuyến hàng sắp tới.

Việc truy xuất thông tin tài xế đang làm việc tại công ty, bao gồm các thông tin: ID, tên tài xế, biển số xe, điểm kỹ năng, điểm đúng giờ, và điểm thái độ.

#### 2. GOM NHÓM CÁC CHUYẾN HÀNG

Từ số lượng tài xế đã chọn ở bước 1, các nhóm chuyến hàng được tạo ra với số lượng nhóm chuyến hàng bằng với số lượng tài xế. *Nhóm chuyến hàng* có thể là chuyến hàng nhập, chuyến hàng xuất hoặc chuyến hàng kết hợp. Người điều hành cần chọn chuyến hàng đưa vào nhóm chuyến hàng sao cho đảm bảo về mặt thời gian không bị trùng lặp trong cùng một nhóm chuyến hàng và điểm cuối của chuyến hàng trước có đoạn đường di chuyển gần với điểm đầu của chuyến hàng sau. Các chuyến hàng chưa cần sắp xếp thì không cần chọn đưa vào các nhóm chuyến hàng. Tóm lại nhóm chuyến hàng có thể là *chuyến hàng kết hợp*, như đã nêu ở mục III.A.

#### 3. SỬ DỤNG SWI-PROLOG ĐỂ SẮP XẾP LỊCH BIỂU CỦA CÁC CHUYẾN HÀNG

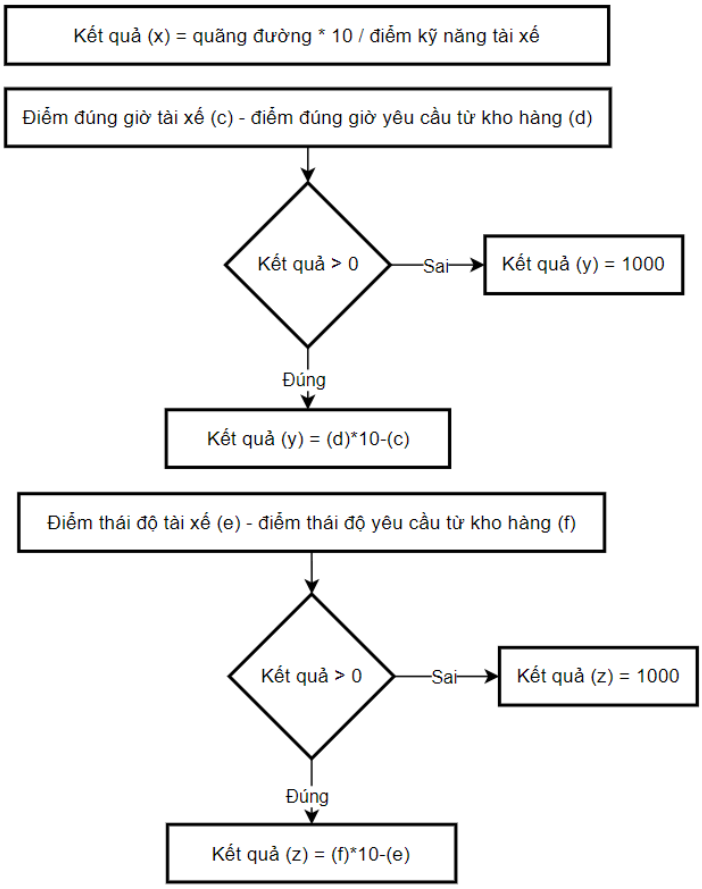
Từ các nhóm chuyến hàng ở bước 2, hệ thống phải tính ra chiều dài các quãng đường, và suy dẫn ra thời gian của mỗi công đoạn trong một nhóm chuyến hàng và và dựa vào những thông tin này, hệ thống có thể xuất ra đoạn mã viết bằng ngôn ngữ SWI-Prolog để sắp xếp lịch biểu cho mỗi chuyến hàng trong một nhóm chuyến hàng. Tức là đoạn mã viết bằng SWI-Prolog có nhiệm vụ giải quyết bài toán con thứ nhất được mô tả ở mục II.C.

Hệ thống có thể truy xuất thông tin về thời gian làm hàng và vị trí của từng địa điểm trong một chuyến hàng được cung cấp bởi khách hàng. Có được địa điểm bắt đầu và kết thúc của một công đoạn trong chuyến hàng, hệ thống phải sử dụng công cụ API Distance Matrix của Goong để suy ra được thời gian, và chiều dài quãng đường di chuyển giữa hai địa điểm. Tóm lại, hệ thống đã tích hợp từ dữ liệu mà công ty có được với các dữ liệu được dẫn xuất ra từ phần mềm API Distance Matrix của Goong.

## 4. SỬ DỤNG SWI-PROLOG ĐỂ GÁN TÀI XẾ VÀO CHUYẾN HÀNG

Mỗi kho hàng (chuyến hàng) và tài xế đều có điểm kỹ năng, điểm thái độ và điểm đúng giờ được tính toán dựa vào công thức được trình bày như trong Hình 2.

Mỗi kho hàng và tài xế khác nhau cho ra những đại lượng (x), (y), (z) khác nhau. Cộng 3 đại lượng *x, y, z* để có được điểm của tài xế với chuyến hàng. Điểm của tài xế với chuyến hàng được dùng để hình thành *ma trận chi phí* (cost matrix) của tài xế và chuyến hàng trong *bài toán gán* (bài toán con thứ 2 được mô tả ở mục II.D).




Hình 2. *Công thức tính điểm để tạo ma trận chi phí cho bước gán tài xế vào chuyến hàng.*

### *D. MÔI TRƯỜNG THỰC NGHIỆM*

Bài toán xếp lịch tài xế được thực hiện trên máy tính với cấu hình: CPU là I7 8750 với 32GB RAM, hệ điều hành: Windows 11. Ngôn ngữ lập trình được sử dụng là: C#, ASP.NET, LINQ, SWI-PROLOG. Các thư viện và API được sử dụng là: CLPFD của SWI-Prolog, Google Map hoặc Goong.

### *E. KẾT QUẢ THỰC NGHIỆM*

Mục đích chính của nghiên cứu này là tự động hóa công tác xếp lịch tài xế tại một công ty vận chuyển hàng hóa. Với Công ty vận chuyển hàng hóa Phan Long, số lượng tài xế của công ty là 8, và số lượng chuyến hàng phải xử lý trong một đợt xử lý trung bình là 10. Để đánh giá hiệu quả của chương trình xếp lịch tài xế, chúng tôi tiến hành 3 lượt thực nghiệm với những đặc điểm khác nhau nhằm mô phỏng các tình huống thực tế mà công ty vận chuyển hàng hóa này thường gặp phải.

#### 1. THỰC NGHIỆM 1

Thực nghiệm này được thực hiện với tình huống: số lượng tài xế là 4 tài xế, số lượng nhóm chuyến hàng là 4 nhóm chuyến hàng (mỗi nhóm chuyến hàng đều chỉ gồm 1 chuyến hàng). Chi tiết về các thông số và kết quả của việc sắp xếp lịch biểu các chuyến hàng được nêu trong Bảng 3. Về bài toán gán tài xế vào chuyến hàng của thực nghiệm 1 gồm có các thông số như sau: số biến quyết định trong bài toán gán là 16 và ma trận chi phí có kích thước $4 \times 4$. Ma trận chi phí của giai đoạn gán tài xế cho chuyến hàng trong thực nghiệm 1 được trình bày trong Bảng 4. Việc tính toán các giá trị trong ma trận này là dựa vào cách tính điểm của tài xế đối với mỗi chuyến hàng được nêu ở Hình 2.

Với thực nghiệm 1, chương trình đã chọn được tổ hợp trị phù hợp cho các biến đạt được chi phí nhỏ nhất với tốc độ thực thi nhanh (chỉ trong 1 giây) và chính xác.

Bảng 3. *Các thông số và kết quả của bước xếp lịch biểu các chuyến hàng trong thực Nghiệm 1*

| Chuyến hàng | Số chặng đường của chuyến hàng | Số biến | Thời lượng tối ưu của chuyến hàng |
|---|---|---|---|
| Chuyến hàng 1 | 7 | 8 | 330 phút |
| Chuyến hàng 2 | 7 | 8 | 230 phút |
| Chuyến hàng 3 | 7 | 8 | 370 phút |
| Chuyến hàng 4 | 7 | 8 | 430 phút |

Bảng 4. *Ma trận chi phí của giai đoạn gán tài xế vào chuyến hàng*

| | | Chuyến hàng | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| Tài xế | 1 | 1125 | 1074 | 217 | 1153 |
| | 2 | 183 | 142 | 211 | 221 |
| | 3 | 235 | 167 | 1208 | 299 |
| | 4 | 216 | 157 | 256 | 270 |

**2. THỰC NGHIỆM 2**

Thực nghiệm này được thực hiện với tình huống: số lượng tài xế là 4 tài xế, số lượng chuyến hàng là 10 chuyến hàng (10 chuyến hàng này được kết hợp thành 4 nhóm chuyến hàng). Số lượng chuyến hàng trên các nhóm chuyến hàng: các nhóm chuyến hàng 1 và 4 gồm có 2 chuyến hàng kết hợp thành, các nhóm chuyến hàng 2 và 3 gồm có 3 chuyến hàng kết hợp thành.

Bảng 5. *Các thông số và kết quả của bước xếp lịch biểu các chuyến hàng trong thực nghiệm 2*

| Chuyến hàng | Số chặng đường của chuyến hàng | Số biến | Thời lượng tối ưu của chuyến hàng |
|---|---|---|---|
| Chuyến hàng 1 | 13 | 14 | 750 phút |
| Chuyến hàng 2 | 19 | 20 | 780 phút |
| Chuyến hàng 3 | 19 | 20 | 840 phút |
| Chuyến hàng 4 | 13 | 14 | 590 phút |

Chi tiết về các thông số và kết quả của việc sắp xếp lịch biểu các chuyến hàng được nêu trong Bảng 5. Về bài toán gán tài xế vào chuyến hàng của thực nghiệm 2 gồm có các thông số như sau: số biến quyết định trong bài toán gán là 16 và ma trận chi phí có kích thước $4 \times 4$.

Với thực nghiệm 2, chương trình đã chọn được tổ hợp trị phù hợp cho các biến, đạt được chi phí nhỏ nhất với tốc độ thực thi nhanh (chỉ trong 3 giây) và chính xác.

**3. THỰC NGHIỆM 3**

Thực nghiệm này nhằm mục đích đo lường sự gia tăng thời gian thực thi của *bài toán gán* (bài toán tối ưu hóa thứ hai) khi qui mô của dữ liệu (số tài xế và số chuyến hàng) gia tăng. Kết quả về thời gian thực thi của bài toán gán thay đổi theo qui mô chuyến hàng được nêu trong Bảng 6.

Kết quả của thực nghiệm 3 cho thấy giải thuật *nhánh-và-cận* được sử dụng trong bài toán gán cũng như bài toán xếp lịch biểu (bài toán tối ưu hóa thứ nhất) gây ra sự gia tăng về thời gian tính toán rất nhanh khi qui mô của dữ liệu gia tăng. Điều này phản ảnh một đặc điểm đáng quan tâm của giải thuật nhánh-và-cận là giải thuật này có độ phức tạp tính toán hàm mũ.

Bảng 6. *Thời gian thực thi của bài toán gán ứng với qui mô dữ liệu*

| Kích thước | Tổng số biến | Thời gian thực thi (giây) |
|---|---|---|
| 4 tài xế, 4 chuyến hàng | 16 | 0.008 |
| 6 tài xế, 6 chuyến hàng | 36 | 0.011 |
| 7 tài xế, 7 chuyến hàng | 49 | 0.097 |
| 8 tài xế, 8 chuyến hàng | 64 | 1.021 |
| 10 tài xế, 10 chuyến hàng | 100 | 10.031 |

Sau khi thực hiện 3 lần thực nghiệm trên dữ liệu thực tế tại Công ty vận chuyển hàng hóa Phan Long, chúng tôi thu được kết quả rất thỏa đáng. Thông qua quá trình hiện thực và thực nghiệm 2 bài toán con (sắp xếp lịch biểu và gán tài xế), chúng tôi có một số nhận xét như sau: i) đoạn mã SWI-Prolog của ngôn ngữ lập trình logic có ràng buộc SWI-Prolog đã giúp tìm ra phương án tối ưu rất nhanh, ii) số nhóm chuyến hàng phải bằng với số tài xế thì chương trình mới có thể xuất ra đoạn mã SWI-Prolog cần xây dựng để thực hiện việc giải bài toán thỏa hệ ràng buộc tối ưu hóa và iii) phương pháp đề xuất khá phù hợp với dữ liệu có qui mô nhỏ và trung bình vì giải thuật nhánh-và cận có độ phức tạp tính toán hàm mũ.

Tóm lại, trong khi giải quyết bài toán xếp lịch tài xế cho Công ty vận chuyển hàng hóa Phan Long một cách thủ công thì tốn thời gian trung bình khoảng 3 giờ để hoàn thành thì sử dụng phương pháp đề xuất dựa vào lập trình ràng buộc chỉ tốn thời gian trung bình khoảng 10 giây.

## IV. KẾT LUẬN

Qua quá trình nghiên cứu và triển khai giải pháp dựa vào lập trình ràng buộc kết hợp với giải thuật nhánh-và-cận cho bài toán xếp lịch tài xế cho Công ty vận chuyển hàng hóa Phan Long, một công ty có qui mô nhỏ và vừa, chúng tôi đã đạt được những kết quả đáng kể. Bằng cách áp dụng lập trình ràng buộc kết hợp với giải thuật nhánh-và-cận và tích hợp dữ liệu từ kho dữ liệu mà công ty có với dữ liệu thời lượng được dẫn xuất ra từ phần mềm Distance Matrix API của Goong, chúng tôi đã tạo ra một hệ thống tự động hóa quy trình xếp lịch, giúp cho công ty tăng cường hiệu xuất làm việc và tăng tính chính xác trong quá trình quản lý công việc vận chuyển hàng. Hệ thống giải quyết hai bài toán con quan trọng trong công tác xếp lịch này: (i) xếp lịch chuyến hàng sao cho tối ưu hóa về thời gian dựa vào *bài toán xếp lịch với các ràng buộc có thứ tự trước sau* và (ii) *bài toán gán* (assignment problem) để phân công tài xế phụ trách các chuyến hàng sao cho thích hợp nhất.

Chúng tôi lựa chọn sự kết hợp lập trình ràng buộc với giải thuật nhánh-và-cận thay vì kết hợp lập trình ràng buộc với giải thuật metaheuristic là do giải thuật nhánh-và-cận là giải thuật tối ưu hóa đem lại lời giải tối ưu chính xác còn các giải thuật metaheuristic chỉ đem lại lời giải xấp xỉ tối ưu. Tuy nhiên giải thuật nhánh-và-cận có độ phức tạp tính toán hàm mũ nên chỉ thích hợp với bộ dữ liệu có quy mô nhỏ và vừa. Từ kết quả thực nghiệm trên dữ liệu thực của công ty Phan Long, chúng tôi rút ra được các kết luận sau đây:

- Giải pháp lập trình ràng buộc kết hợp giải thuật nhánh-và-cận giúp giảm thiểu thời gian xếp lịch và tăng cường tính linh hoạt trong việc thích ứng với các yêu cầu của khách hàng về thái độ, tính đúng giờ và kỹ năng của tài xế.
- Hệ thống đã cho thấy sự hiệu quả trong việc tối ưu hóa sử dụng tài nguyên (tài xế, thời gian) và giảm chi phí hoạt động của công ty.
- Các kết quả thử nghiệm cho thấy tính ổn định, tính hữu hiệu về thời gian tính toán và tính chính xác của giải pháp, đồng thời khẳng định tiềm năng ứng dụng rộng rãi giải pháp trong ngành vận chuyển và logistics.

Tuy nhiên, nghiên cứu này chỉ mới thực hiện việc xếp lịch tài xế cho một công ty vận chuyển hàng hóa có quy mô nhỏ và vừa. Do đó, chúng tôi dự định thử áp dụng phương pháp đề xuất cho bài toán xếp lịch tài xế tại một công ty vận chuyển hàng hóa có quy mô lớn hơn để xem giải thuật tối ưu hóa nhánh-và-cận thích nghi được với bộ dữ liệu có quy mô lớn đến cỡ nào. Kế đến, chúng tôi dự định so sánh tính hữu hiệu về thời gian thực thi của phương pháp đề xuất (sử dụng giải thuật nhánh và cận) với phương pháp dựa vào lập trình ràng buộc kết hợp sử dụng một giải thuật metaheuristic, giải thuật di truyền. Ngoài ra, chúng tôi dự định phát triển các tính năng mở rộng trong hệ thống như tích hợp GPS để theo dõi vị trí tài xế và cập nhật lịch trình hoạt động theo thời gian thực.

## V. LỜI CẢM ƠN

Các tác giả của nghiên cứu này chân thành cảm ơn Công ty Vận chuyển hàng hóa Phan Long, Quận 11, Tp. Hồ Chí Minh đã cho phép sử dụng bộ dữ liệu xếp lịch tài xế của công ty.

## VI. TÀI LIỆU THAM KHẢO


[1] M. Koubaa, S. Dhouib, D. Dhouib, A. El-Mhamedi (2016). Truck Driver Scheduling Problem: Literature Review. *IFAC-PapersOnline* **49**:12, pp. 1950-1955.

[2] A. Azadeh, M. H. Farahani, H. Eivazy, S. N. Shirkouhi, G. Asadipour (2013). A Hybrid Meta-heuristic Algorithm for Optimization of Crew Scheduling. *Applied Soft Computing*. **13**, pp. 158-164.

[3] Z. Tian, Q. Song (2013). Modeling and Algorithm of the Crew Scheduling Problem on High-speed Railway Lines, *Procedia- Social and Behavioral Sciences* **96**, pp. 1443-1452.

[4] A. F. Han, E. C. Li (2014). A Constraint Programming-based Approach to the Crew Scheduling Problem of the Taipei Mass Rapid Transit System. *Ann. Oper. Res.* **223**, pp. 173-193.

[5] M. Jacyna, M. Izdebski (2014). The Ant Algorithm for Solving the Assignment of Vehicles to Tasks in the Municipal services companies. *Journal of KONES Powertrain and Transport* **21**:2, pp. 113-119.

[6] M. Izdebski, M. Jacyna (2014). Some Aspects of the Application of Genetic Algorithm for Solving the Assignment Problem of Tasks to Resources in a Transport Company. *Logistic and Transport* **21**: 1, pp. 13-20, Wroclaw.

[7] M. Jacyna, M. Izdebski, E. Szczepanski, P. Golda (2018). The Task Assignment of Vehicles for a Production Company. *Symmetry* **10**: 551.

[8] C. A. Cruz, P. Munari, R. Morabito (2020). A Branch-and-Price method for the Vehicle Allocation Problem. *Computers & Industrial Engineering* **149**, Article 106745, November.

[9] D. Benli, A. Aydin, M. Cimen, M. Soysal (2022). Analyses on the Effect of Time Window Choices on Sustainable Vehicle Allocation Problems. *Journal of the Faculty of Economics and Administrative Sciences*, **12**: 1, pp. 157-175.

[10] F. Rossi, P. van Beek, T. Walsh (2008). Chapter 4: Constraint Programming. *Foundations of Artificial Intelligence* **3**, pp. 181-211.

[11] M. Dincbas, H. Simonis, P. van Hentenryck (1990). Solving large combinatorial problems in Logic Programming. *J. Logic Programming* **8**, pp. 7-93.

[12] A. Levitin (2012). *Introduction to the Design and Analysis of Algorithms*-3rd Edition, Pearson, 2012.

[13] Dương Tuấn Anh (2023). Bài giảng chương "Lập Trình Logic có Ràng Buộc", Môn cao học Lập Trình Logic và Ràng Buộc, Khoa Khoa Học Và Kỹ Thuật Máy Tính, Trường Đại Học Bách Khoa, Đại Học Quốc Gia TP.Hồ Chí Minh.

[14] A. Niederinski (2014). *A Gentle Guide to Constraint Logic Programming via ECL$^i$Pse*, 3rd Edition, Jacek Skalmierski Computer Studio, 2014.

[15] Microsoft, Support of Microsoft. https://support.microsoft.com/vi-vn/topic/31846479-c656-f2a4-bc24-c9803a97e62c (accessed April 15, 2024).

[16] SWI-Prolog 7.6.0 Reference Manual. https://www.swi-prolog.org/download/stable/doc/SWI-Prolog-7.6.0.pdf (accessed April 15, 2024).

[17] J. Wielemaker (2012). *SWI-Prolog 6.0 Reference Manual*, University of Amsterdam, The Netherlands, March.

[18] J. Wielemaker (2014). SWI-Prolog version 7 Extensions. *Proc. of International Joint Workshop on Implementation of Constraint and Logic Programming and Logic-based Methods in Programming Environment*, pp. 109-124.


# APPLYING CONSTRAINT-PROGRAMMING TO DRIVER SCHEDULING IN A TRANSPORT COMPANY

**Duong Tuan Anh, Nguyễn Đức Huy**

**ABSTRACT**— Driver scheduling is a crucial problem which affects the performance of a freight transport company and its customer satisfaction. The duties of drivers include punctual load delivery, satisfying road transportation rules and warehouse regulations. Therefore, driver scheduling problem is not only a human resource assignment problem but also an optimization task for the whole freight transport system. In order to apply constraint programming approach to a case study: driver scheduling in Phan Long Transport Company, District 11, Ho Chi Minh City, we use constraint programming approach combined with branch-and-bound algorithm to solve two subproblems: i) scheduling with precedence constraints and ii) assignment problem to driver scheduling. Experimental results on the real-world data at Phan Long Company show that the proposed approach brings out not only robust and effective solutions but also the time efficiency in solving process.

**Keywords**— driver scheduling, freight transportation, constraint programming, branch-and-bound algorithm, scheduling problem with precedence constraints, assignment problem.

**Dương Tuấn Anh** tốt nghiệp tiến sĩ ngành khoa học máy tính tại Học Viện Công nghệ Á Châu (Asian Institute of Techology), Bangkok, Thái Lan, năm 1998 và đó cũng là nơi ông tốt nghiệp thạc sĩ với cùng chuyên ngành. Ông đã là Phó Giáo Sư tại khoa Khoa học và Kỹ thuật máy tính, trường Đại Học Bách Khoa, ĐHQG Tp. Hồ Chí Minh từ năm 2007. Từ năm 2020 đến nay, ông là giảng viên khoa Công nghệ thông tin, trường Đại học Ngoại ngữ-Tin học Tp. Hồ Chí Minh. Lĩnh vực nghiên cứu chính của ông là metaheuristics, học máy và khai phá dữ liệu chuỗi thời gian. Ông là đồng tác giả của trên 138 bài báo khoa học.

**Nguyễn Đức Huy** tốt nghiệp cử nhân ngành Công nghệ thông tin của chương trình liên kết quốc tế giữa đại học Greenwich, Vương Quốc Anh với Đại học FPT năm 2021 và tốt nghiệp thạc sĩ ngành Công nghệ thông tin tại Trường Đại học Ngoại ngữ- Tin học Tp. Hồ Chí Minh năm 2024. Anh là chuyên viên Công nghệ thông tin của Công yy Vận chuyển Phan Long, Quận 11, Tp. Hồ Chí Minh từ 2021 đến nay. Lĩnh vực nghiên cứu chính của anh là ứng dụng AI vào logistics.