ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Đình Tân

PHÁT TRIỂN THUẬT TOÁN XẤP XỈ CHO BÀI TOÁN TỐI ĐA HÀM SUBMODULAR

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

Hà Nội – 2026

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Trần Đình Tân

PHÁT TRIỂN THUẬT TOÁN XẤP XỈ CHO BÀI TOÁN TỐI ĐA HÀM SUBMODULAR

Ngành đào tạo: Khoa học máy tính
Mã số: 9480101

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

NGHIÊN CỨU SINH **CÁN BỘ HƯỚNG DẪN**

Trần Đình Tân **PGS.TS. Hoàng Xuân Huấn**

PGS.TS. Phạm Văn Cảnh

XÁC NHẬN CỦA ĐƠN VỊ ĐÀO TẠO

Hà Nội – 2026

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
DrMSC	DR-submodular Maximization under Size Constraint	Tối đa DR-submodular với ràng buộc lực lượng
IC	Independent Cascade	Bậc độc lập
IM	Influence Maximization	Tối đa ảnh hưởng
kIMIK	k-topic Influence Maximization Under Individual Knapsack Constraint	Tối đa ảnh hưởng theo k-chủ đề với ràng buộc chi phí nhóm
kPMIK	k-type Product Revenue Maximization Under Individual Knapsack Constraint	Tối đa doanh thu sản phẩm theo k-loại với ràng buộc chi phí nhóm
kSMIK	k-Submodular Maximization under Individual Knapsack Constraints	Tối đa hàm k-submodular với ràng buộc chi phí nhóm
kSPIK	k-type Sensor Placement under Individual Knapsack constraint	Bố trí cảm biến theo k-loại với ràng buộc chi phí nhóm
LT	Linear Threshold	Ngưỡng tuyến tính
MSC	Minimum cost Submodular Cover	Bài toán tập phủ với chi phí nhỏ nhất
SMC	Submodular Maximization under Cardinality Constraint	Tối đa hàm submodular với ràng buộc lực lượng
SMK	Submodular Maximization under Knapsack Constraint	Tối đa hàm submodular với ràng buộc chi phí

DANH SÁCH BẢNG

DANH MỤC CÁC KÝ HIỆU TOÁN HỌC

Ký hiệu	Ý nghĩa trong luận án
V	Tập cơ sở hữu hạn có n phần tử, $V = \{e_1, e_2, \ldots, e_n\}$
n	Số phần tử của tập cơ sở V
2^V	Không gian tất cả các tập con của tập cơ sở V
$(k+1)^V$	Họ k tập đôi một không giao nhau, được gọi là k-tập (k-set), cụ thể: $(k+1)^V = \{(S_1, S_2, \ldots, S_k) \mid S_i \subseteq V,\ \forall i \in [k],\ S_i \cap S_j = \emptyset,\ \forall i \neq j\}$
$\mathbb{Z}_+^V$	Lưới nguyên không âm trên tập cơ sở V, tức tập các vectơ nguyên không âm được đánh chỉ số bởi V
$\mathbb{R}$	Tập hợp các số thực
$\mathbb{R}_+$	Tập hợp các số thực không âm
ϵ	Tham số chính xác của các thuật toán xấp xỉ
$O(\cdot)$	Độ phức tạp của thuật toán
$\tilde{O}(\cdot)$	Độ phức tạp của thuật toán khi bỏ qua các yếu tố logarit
$f(e \mid S)$	Lợi ích biên thêm phần tử e vào tập S, cụ thể $f(e \mid S) = f(S \cup e) - f(S)$
$supp$	$supp(\mathbf{x}) = \cup_{i \in [k]} X_i$, trong đó X_i là tập thứ i của $\mathbf{x} = (X_1, X_2, \ldots, X_k) \in (k + 1^V)$
$supp_i$	$supp_i(\mathbf{x}) = X_i$, trong đó X_i là tập thứ i của $\mathbf{x} = (X_1, X_2, \ldots, X_k) \in (k + 1^V)$
$\Delta_{(e,i)} f(\mathbf{x})$	Lợi ích biên khi thêm $e \in V$ vào tập thứ i của $\mathbf{x} \in (k+1)^V$, cụ thể: $\Delta_{(e,i)} f(\mathbf{x}) = f(\mathbf{x} \sqcup (e, i)) - f(\mathbf{x})$
$\mathbf{x}(e)$	Giá trị tọa độ của $e \in V$ trong vector $\mathbf{x} \in \mathbb{Z}_+^V$
$\|\mathbf{x}\|_1$	Chuẩn ℓ_1 của vectơ $\mathbf{x} \in \mathbb{Z}^V$, được định nghĩa bởi $\|\mathbf{x}\|_1 = \sum_{e \in V} \mathbf{x}(e)$
$\mathbf{1}_e$	Vectơ đơn vị $\mathbf{1}_e \in \mathbb{Z}_+^V$ được xác định bởi $\mathbf{1}_e(t) = 1$ nếu $t = e$ và $\mathbf{1}_e(t) = 0$ nếu $t \neq e$
$f(\mathbf{y} \mid \mathbf{x})$	Lợi ích biên của hàm dr-submodular f khi cộng thêm $\mathbf{y}$ vào $\mathbf{x}$, cụ thể: $f(\mathbf{y} \mid \mathbf{x}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$
$\{\mathbf{x}\}$	Là tập hợp các phần tử $e \in V$ sao cho $\mathbf{x}(e) > 0, \forall \mathbf{x} \in \mathbb{Z}_+^V$

LỜI MỞ ĐẦU

1. Bối cảnh nghiên cứu

Trong bối cảnh khoa học máy tính hiện đại, các bài toán tối ưu tổ hợp ngày càng giữ vai trò then chốt trong nhiều lĩnh vực ứng dụng quan trọng như học máy, khai phá dữ liệu và hệ thống khuyến nghị [32, 110]. Một bài toán tối ưu tổ hợp có thể được phát biểu như sau: Cho một tập hợp hữu hạn Ω gọi là tập nghiệm khả thi (hoặc trạng thái chấp nhận được) và một hàm mục tiêu $f : \Omega \to \mathbb{R}$. Mục tiêu của bài toán tối ưu tổ hợp là:

$$s^* = \arg\max(\min)_{s\in\Omega} f(s),$$

tùy theo bài toán là cực đại hay cực tiểu [62].

Mục tiêu của bài toán là lựa chọn một phần tử tối ưu từ Ω sao cho giá trị hàm mục tiêu đạt cực đại (hoặc cực tiểu), trong đó Ω phải là một tập hữu hạn (hoặc đếm được). Đây là đặc trưng của các bài toán tối ưu tổ hợp, phân biệt với các bài toán tối ưu rời rạc có không gian nghiệm vô hạn. Dễ dàng thấy rằng, hàm mục tiêu đóng vai trò trung tâm trong tối ưu tổ hợp, vì nó không chỉ định hướng việc lựa chọn lời giải mà còn ảnh hưởng đến tính chất tính toán và khả năng thiết kế thuật toán hiệu quả. Một số hàm mục tiêu thường gặp trong tối ưu tổ hợp cụ thể như: tổng lợi ích/giá trị (điển hình trong bài toán cái túi); chi phí quãng đường tối thiểu (trong bài toán người bán hàng); cũng như số lượng hoặc dung lượng tối thiểu cần dùng (ví dụ như trong bài toán bin-packing) [101];...

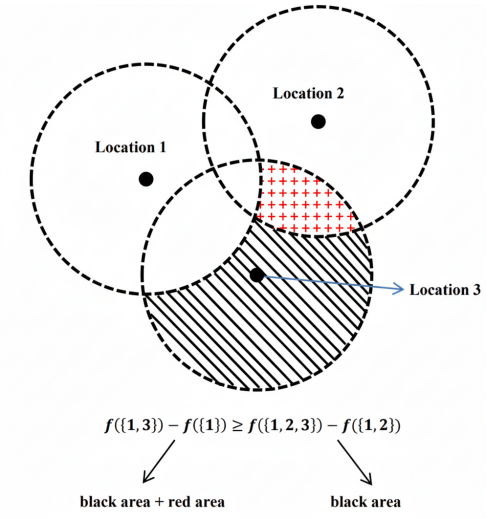
Một trong những hướng tiếp cận nổi bật trong việc giải quyết các bài toán tối ưu tổ hợp là khai thác cấu trúc đặc biệt của hàm mục tiêu. Trong số đó, hàm *submodular* – với tính chất "lợi ích biên giảm dần" – đã thu hút nhiều sự quan tâm trong cộng đồng nghiên cứu nhờ khả năng cho phép thiết kế các thuật toán xấp xỉ hiệu quả, đi kèm với đảm bảo lý thuyết vững chắc. Cụ thể, một hàm tập hợp $f : 2^V \to \mathbb{R}$ (ánh xạ từ không gian tất cả các tập con của V sang tập số thực) được gọi là submodular [94] nếu với mọi $A \subseteq B \subseteq V$ và $e \in V \setminus B$, ta có:

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B).$$

Tính chất này phản ánh hiện tượng lợi ích biên giảm dần (diminishing returns), một đặc trưng thường thấy trong các hệ thống, nơi các yếu tố như sự phối hợp, ảnh hưởng giữa các thành phần đóng vai trò quan trọng.

Một ví dụ tiêu biểu thể hiện vai trò của hàm submodular là bài toán đặt cảm biến trong mạng giám sát [70]. Giả sử tập cơ sở V biểu diễn tập các vị trí tiềm năng trong không gian mà tại đó các cảm biến có thể được lắp đặt. Khi đặt một cảm biến tại một vị trí cụ thể, nó bao phủ một vùng không gian hình tròn xung quanh, như minh họa trong Hình 0.1. Gọi $f(S)$ là tổng diện tích được bao phủ khi lắp đặt cảm biến tại các vị trí

thuộc tập con $S \subseteq V$. Khi đó, hàm f là một hàm submodular, phản ánh tính chất lợi ích biên giảm dần. Tính chất này được thể hiện rõ trong Hình 0.1: lợi ích tăng thêm thu được (*vùng gạch chéo + vùng dấu cộng*) khi thêm cảm biến thứ ba sau khi đã đặt cảm biến thứ nhất lớn hơn lợi ích tăng thêm thu được (*chỉ vùng gạch chéo*) khi thêm cảm biến thứ ba vào một hệ thống đã có sẵn hai cảm biến thứ nhất và thứ hai.



Hình 0.1: Ví dụ về ý nghĩa của hàm submodular với bài toán đặt cảm biến [88]

Với tính biểu đạt linh hoạt và khả năng mô hình hóa các hiện tượng thực tiễn có tính tương tác, hàm submodular đã được ứng dụng rộng rãi trong nhiều lĩnh vực. Trong học máy, các bài toán chọn đặc trưng sử dụng hàm submodular để cân bằng giữa thông tin và tính đa dạng của tập thuộc tính. Trong phân tích mạng xã hội, hàm submodular mô tả hiệu quả quá trình lan truyền ảnh hưởng trong các mô hình tuyến tính hoặc ngưỡng độc lập [66]. Trong các hệ thống cảm biến, nó phản ánh giá trị thông tin thu thập được tại các vị trí giám sát khác nhau [70]. Ngoài ra, các ứng dụng như tóm tắt văn bản, phân bổ ngân sách và tối ưu tài nguyên cũng thường được mô hình hóa thông qua hàm submodular [78, 91]. Điểm mạnh nổi bật của tối ưu hàm submodular không chỉ đến từ tính ứng dụng thực tiễn, mà còn ở khả năng thiết kế các thuật toán hiệu quả với đảm bảo lý thuyết rõ ràng. Trong nhiều trường hợp, mặc dù bài toán tối ưu submodular là NP-khó[1], nhưng vẫn có thể xây dựng các thuật toán xấp xỉ đạt hệ số xấp xỉ tối ưu[2], ví dụ như thuật toán tham lam đạt tỉ lệ $1 - \frac{1}{e} \approx 0.63$ trong trường hợp hàm submodular đơn điệu với ràng buộc lực lượng [93]. Do đó, tối ưu submodular là một trong những trường hợp hiếm hoi giao thoa giữa lý thuyết vững chắc và hiệu quả thực tế trong tối ưu tổ hợp hiện đại.

Phần lớn các bài toán trong tối ưu tổ hợp là NP-khó [124]. Nhiều phương pháp đã được đề xuất nhằm tìm nghiệm hiệu quả trong thực tế, cụ thể: (1) Thuật toán chính

[1] NP-khó là lớp bài toán chưa có thuật toán đa thức nào được biết để giải chính xác mọi trường hợp.
[2] Hệ số xấp xỉ tối ưu của một bài toán là hệ số xấp xỉ tốt nhất có thể đạt được, trừ khi $P = NP$.

xác như quy hoạch động, nhánh-cận hoặc ràng buộc-cắt đảm bảo tìm được nghiệm tối ưu, nhưng chi phí tính toán cao và chỉ áp dụng được cho bài toán có kích thước đầu vào nhỏ hoặc có cấu trúc đặc biệt [4]. (2) Thuật toán xấp xỉ là hướng tiếp cận có bảo đảm lý thuyết về chất lượng lời giải trong thời gian đa thức theo kích thước dữ liệu đầu vào. Tiêu biểu như thuật toán Christofides cho bài toán người bán hàng với tỉ lệ xấp xỉ $\frac{1}{1.5} \approx 0.67$ [24], hoặc thuật toán tham lam đạt $1 - \frac{1}{e} \approx 0.63$ cho bài toán tối ưu submodular đơn điệu [93]. Phương pháp này nổi bật ở khả năng cân bằng giữa độ chính xác và hiệu quả tính toán. (3) Heuristic và metaheuristic, như tham lam, tìm kiếm lân cận, hoặc thuật toán ngẫu nhiên, cho nghiệm nhanh, hiệu quả trong thực nghiệm nhưng thiếu bảo đảm lý thuyết [11, 57, 67]. (4) Nới lỏng bài toán bằng quy hoạch tuyến tính (LP) hoặc nhân tử Lagrange giúp xây dựng một bài toán dễ giải hơn, từ đó cung cấp cận cho nghiệm tối ưu hoặc định hướng lựa chọn phần tử. Cách làm này hỗ trợ các thuật toán nhánh-cận hoặc thiết kế thuật toán xấp xỉ hiệu quả hơn [45]. (5) Học máy gần đây được ứng dụng để sinh lời giải hoặc dự đoán lời giải, cho kết quả tốt với dữ liệu quen thuộc nhưng thiếu bảo đảm lý thuyết và cần huấn luyện công phu [9, 10].

Từ những đánh giá trên, có thể thấy rằng trong các bài toán tối ưu tổ hợp với kích thước dữ liệu lớn, khi thuật toán chính xác không khả thi và heuristic thiếu bảo đảm lý thuyết, thuật toán xấp xỉ nổi bật như sự lựa chọn phù hợp, đặc biệt khi bài toán có cấu trúc hàm mục tiêu thuận lợi như submodular. Chính vì vậy, hướng tiếp cận của nghiên cứu sinh là tập trung đề xuất các thuật toán xấp xỉ cho bài toán tối ưu hàm submodular và cụ thể tên luận án: "*Phát triển thuật toán xấp xỉ cho bài toán tối đa hàm submodular*".

2. Tổng quan vấn đề nghiên cứu

Trong lĩnh vực tối ưu tổ hợp, các bài toán liên quan đến hàm submodular xuất hiện với nhiều dạng bài toán khác nhau, nổi bật là các bài toán tối đa và đối ngẫu của bài toán tối đa – bài toán phủ submodular. Mỗi bài toán này đều có đặc trưng riêng về không gian nghiệm, hàm mục tiêu và mục tiêu tối ưu, từ đó dẫn đến sự đa dạng trong các kỹ thuật giải quyết. Việc phân loại và phân tích các bài toán này theo mục tiêu tối ưu giúp định hình rõ ràng phạm vi và định hướng cho việc phát triển các thuật toán.

Cùng với sự phát triển của lý thuyết hàm submodular, nhiều biến thể mở rộng của hàm submodular đã được đề xuất nhằm mô tả tốt hơn các bài toán thực tế có cấu trúc phức tạp. Hai lớp hàm tiêu biểu là k-submodular, được sử dụng cho các bài toán phân hoạch nhiều nhóm và DR-submodular, cho phép mở rộng miền xác định sang lưới nguyên. Các mở rộng này giúp mô hình hóa các bài toán trong thực tế chính xác hơn, tuy nhiên, kéo theo việc phải giải quyết các bài toán tối ưu mới, đòi hỏi thiết kế các thuật toán chuyên biệt tương ứng.

Bên cạnh việc mở rộng miền xác định, các bài toán tối ưu hàm submodular còn chịu ảnh hưởng mạnh bởi các đặc trưng của hàm mục tiêu như tính đơn điệu và các ràng

buộc đi kèm. Trong thực tế, bài toán có thể bị chi phối bởi các ràng buộc như ràng buộc chi phí (knapsack), ràng buộc lực lượng (cardinality), hoặc ràng buộc cấu trúc (matroid), ... Khi kết hợp với các tính chất của hàm mục tiêu (đơn điệu hoặc không đơn điệu), các ràng buộc này tạo ra hàng loạt bài toán, ví dụ như: tối đa hàm submodular không đơn điệu với ràng buộc chi phí, tối đa hàm k-submodular không đơn điệu với ràng buộc lực lượng, . . .

Dù đã có nhiều nỗ lực nghiên cứu nhằm xây dựng các thuật toán xấp xỉ hiệu quả, nhưng thực tế vẫn còn nhiều bài toán chưa có lời giải thỏa đáng *(Xem bảng 0.2)*. Một số bài toán đã được đề xuất thuật toán nhưng hệ số xấp xỉ còn thấp hoặc độ phức tạp tính toán chưa tốt, chưa đáp ứng được yêu cầu thực tiễn. Điều này thể hiện rõ sự thiếu hụt trong hệ thống lý thuyết hiện tại, đặc biệt là khi xem xét các trường hợp bài toán phức tạp hơn với hàm không đơn điệu và nhiều ràng buộc đồng thời.

Bảng 0.2: Ma trận một số bài toán tiêu biểu tối ưu hàm submodular.

Lớp bài toán tối đa hàm submodular						
Ràng buộc	Sub. đơn điệu	Sub. không đơn điệu	k-Sub. đơn điệu	k-Sub. không đơn điệu	DR-Sub. đơn điệu	DR-Sub. không đơn điệu
Không ràng buộc	–	✓[21]	✓[63]	✓[100]	–	✓[96]
Lực lượng	✓[94]	✓[15]	✓[98]	✓[126]	✓[113]	? **Bài toán 3**
Chi phí	✓[116]	✓[53] **Bài toán 1**	✓[20]	✓[51]	✓[117]	?
Matroid	✓[17]	✓[43]	[105]	[115]	?	?
Chi phí nhóm	?	?	?	? **Bài toán 2**	?	?
Lớp bài toán phủ submodular						
Lực lượng tối thiểu	✓[125]	✓[26]	✓[95]	?	?	?
Chi phí tối thiểu	✓[26] **Bài toán 4**	✓[26]	?	?	?	?

✓ Các bài toán đã có thuật toán đề xuất

– Các bài toán tầm thường không cần đề xuất thuật toán

? Các bài toán chưa có thuật toán đề xuất

Bài toán 1,2,3,4 là bốn bài toán nghiên cứu của luận án. *Sub.* là viết tắt của *submodular.*
Ghi chú: Các công trình được tham chiếu trong ma trận chỉ mang tính đại diện tiêu biểu cho từng bài toán và chưa phản ánh đầy đủ toàn bộ các nghiên cứu liên quan.

Từ ma trận phân loại trong Bảng 0.2, có thể nhìn thấy rõ vị trí của bốn bài toán nghiên cứu trong bức tranh chung của tối ưu hàm submodular. Bảng được tổ chức theo ba trục chính: loại hàm mục tiêu, tính đơn điệu và dạng ràng buộc. Các ô màu xanh lá thể hiện những lớp bài toán đã có thuật toán tương ứng; các ô màu vàng thể hiện những

trường hợp cần tiếp tục nghiên cứu. Do đó, bảng không chỉ tóm tắt các lớp bài toán tiêu biểu, mà còn làm rõ những khoảng trống mà luận án hướng tới.

Trên cơ sở đó, mạch nghiên cứu của luận án được xây dựng từ bài toán nền tảng đến các biến thể mở rộng và bài toán đối ngẫu. Bài toán nghiên cứu 1, tức bài toán tối đa hàm submodular với ràng buộc chi phí, là điểm xuất phát trong lớp hàm submodular cổ điển. Bài toán nghiên cứu 2 và bài toán nghiên cứu 3 mở rộng hướng nghiên cứu này sang hai lớp hàm tổng quát hơn, lần lượt là hàm k-submodular với ràng buộc chi phí nhóm và hàm DR-submodular với ràng buộc lực lượng. Bài toán nghiên cứu 4, tức bài toán phủ submodular với chi phí tối thiểu, là bài toán đối ngẫu của bài toán nghiên cứu 1: thay vì tối đa hóa giá trị hàm dưới ràng buộc chi phí, bài toán này tìm tập có chi phí nhỏ nhất để đạt một ngưỡng giá trị cho trước.

Với cách tổ chức như vậy, luận án tiếp cận vấn đề theo hai hướng bổ sung cho nhau. Đối với các lớp bài toán đã có nền tảng nghiên cứu, cụ thể là bài toán nghiên cứu 1 và bài toán 4, mục tiêu là cải thiện hiệu quả thuật toán thông qua hệ số xấp xỉ, độ phức tạp truy vấn hoặc khả năng triển khai thực nghiệm. Đối với các lớp bài toán mở rộng, cụ thể là bài toán nghiên cứu 2 và bài toán nghiên cứu 3, mục tiêu là phát triển thuật toán xấp xỉ có bảo đảm lý thuyết cho những bối cảnh ràng buộc phức tạp hơn. Sự kết hợp này giúp luận án vừa kế thừa các kết quả nền tảng, vừa mở rộng phạm vi nghiên cứu sang các lớp hàm và ràng buộc mới.

Từ các khoảng trống nêu trên, luận án tập trung vào bốn câu hỏi nghiên cứu chính. Thứ nhất, đối với bài toán tối đa hàm submodular với ràng buộc chi phí, làm thế nào để thiết kế thuật toán có bảo đảm xấp xỉ tốt hơn hoặc có độ phức tạp truy vấn thấp hơn so với các hướng tiếp cận đã có? Thứ hai, đối với bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm, có thể xây dựng thuật toán xấp xỉ với bảo đảm lý thuyết cho lớp bài toán mở rộng này hay không? Thứ ba, đối với bài toán tối đa hàm DR-submodular với ràng buộc lực lượng, có thể khai thác cấu trúc lợi ích biên giảm dần trên lưới nguyên để thiết kế thuật toán hiệu quả hay không? Thứ tư, đối với bài toán 4, có thể cải thiện hiệu quả thuật toán cho bài toán phủ submodular với chi phí tối thiểu trong khi vẫn bảo đảm chất lượng lời giải hay không? Các câu hỏi này tạo thành mạch nghiên cứu xuyên suốt của luận án.

3. Bài toán nghiên cứu

Dựa trên những khoảng trống trong lý thuyết và nhu cầu thực tiễn đã trình bày ở phần trước, luận án tập trung nghiên cứu bốn bài toán đại diện trong lớp tối ưu hàm submodular và các mở rộng của nó. Cụ thể:

- **Bài toán nghiên cứu 1:** Bài toán tối đa hàm submodular với ràng buộc chi phí. Một cách ngắn gọn bài toán được phát biểu như sau: Cho một tập cơ sở V có n phần tử, một hàm submodular $f : 2^V \to \mathbb{R}_+$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí

của một tập con $S \subseteq V$ là $c(S) = \sum_{e \in S} c(e)$. Với một ngưỡng chi phí $B > 0$, mục tiêu là tìm một tập con $S \subseteq V$ sao cho: $f(S)$ lớn nhất và $c(S) \leq B$. Bài toán nghiên cứu 1 là bài toán tổng quát cho nhiều ứng dụng như chọn đặc trưng với chi phí, phân bổ tài nguyên, hoặc tối đa ảnh hưởng trong mạng xã hội có giới hạn ngân sách. Hàm mục tiêu tổng quát gây ra khó khăn lớn trong thiết kế thuật toán.

- **Bài toán nghiên cứu 2:** Bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm. Bài toán là mở rộng tự nhiên của bài toán tối ưu submodular sang không gian $(k+1)^V$ – là không gian của họ k tập đôi một không giao nhau. Bài toán có nhiều ứng dụng trong phân cụm, phân bổ đa nhãn và ra quyết định trong các hệ thống đa thành phần. Việc đưa thêm ràng buộc chi phí nhóm làm tăng độ phức tạp đáng kể của bài toán. Một cách ngắn gọn bài toán được phát biểu như sau: Cho một tập cơ sở V, một hàm k-submodular $f : (k+1)^V \to \mathbb{R}_+$ xác định trên không gian họ k tập đôi một không giao nhau, được gọi là k-tập (k-set), cụ thể: $(k+1)^V = \{(S_1, S_2, \ldots, S_k) \mid S_i \subseteq V,\ \forall i \in [k],\ S_i \cap S_j = \emptyset,\ \forall i \neq j\}$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Với mỗi vị trí $i \in [k] = \{1, 2, \ldots, k\}$, tổng chi phí của tập S_i được định nghĩa là $c_i(\mathbf{s}) = \sum_{e \in S_i} c(e)$ và mỗi vị trí có một ngân sách riêng $B_i > 0$. Bài toán yêu cầu tìm một k-tập $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ sao cho: $c_i(\mathbf{s}) \leq B_i$, với mọi $i \in [k]$ và giá trị hàm mục tiêu $f(\mathbf{s})$ được tối đa.

- **Bài toán nghiên cứu 3:** Bài toán tối đa hàm DR-submodular với ràng buộc lực lượng. Bài toán là mở rộng tự nhiên của bài toán tối ưu submodular sang không gian $\mathbb{Z}_+^V$ – là lưới nguyên không âm trên tập cơ sở V. Bài toán này xuất hiện trong các mô hình có thể phân bổ nhiều lần, nhiều cấp. Một cách ngắn gọn bài toán được phát biểu như sau: Cho một tập cơ sở V, một hàm DR-submodular $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$ (ánh xạ từ lưới nguyên không âm trên tập cơ sở V đến tập số thực không âm), một lưới nguyên bị chặn bởi vectơ $\mathbb{B} \in \mathbb{Z}_+^V$ và một số nguyên dương $k > 0$. Bài toán yêu cầu tìm vectơ $\mathbf{x} \leq \mathbb{B}$ sao cho tổng kích thước $\|\mathbf{x}\|_1 \leq k$ và giá trị $f(\mathbf{x})$ đạt cực đại. Việc tối ưu hàm DR-submodular đặt ra thách thức lớn cả về biểu diễn lời giải và phân tích hiệu năng.

- **Bài toán nghiên cứu 4:** Bài toán phủ submodular với chi phí tối thiểu là đối ngẫu của bài toán tối đa hàm submodular với ràng buộc chi phí (Bài toán 1). Thay vì tối đa hàm mục tiêu, mục tiêu là tìm tập con nhỏ nhất (về mặt chi phí) sao cho giá trị hàm đạt một ngưỡng yêu cầu. Một cách ngắn gọn bài toán được phát biểu: Cho một tập cơ sở V, một hàm submodular đơn điệu $f : 2^V \to \mathbb{R}_+$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí của một tập con $S \subseteq V$ là $c(S) = \sum_{e \in S} c(e)$. Với một ngưỡng chất lượng $T > 0$, mục tiêu là tìm một tập con $S \subseteq V$ sao cho: $f(S) \geq T$ và $c(S)$ là nhỏ nhất. Bài toán này phản ánh các tình huống như bảo đảm chất lượng dịch vụ, bao phủ tập dữ liệu hoặc thiết lập hệ thống giám sát hiệu quả.

Luận án đặt mục tiêu phát biểu bài toán một cách chặt chẽ, thiết kế các thuật toán xấp xỉ mới có đảm bảo lý thuyết và triển khai thực nghiệm trên các tập dữ liệu có ý

nghĩa thực tiễn. Mỗi bài toán được phân tích riêng biệt, nhưng đồng thời góp phần hình thành một khung lý thuyết chung cho tối ưu submodular mở rộng với ràng buộc.

4. Mục tiêu và đóng góp của luận án

Luận án hướng đến việc xây dựng các thuật toán tối ưu hiệu quả cho các bài toán submodular và các mở rộng của nó trong những bối cảnh phức tạp hơn. Với cách tiếp cận toàn diện, kết hợp giữa lý thuyết và thực nghiệm, các mục tiêu và đóng góp của luận án có thể được phân loại như sau:

Về mặt lý thuyết: (1) Đề xuất các thuật toán xấp xỉ mới cho từng bài toán cụ thể, với thiết kế phù hợp cho từng loại hàm mục tiêu như submodular, k-submodular hoặc DR-submodular; (2) Phân tích chặt chẽ tỉ lệ xấp xỉ đạt được và độ phức tạp tính toán của các thuật toán, từ đó cung cấp đảm bảo lý thuyết rõ ràng cho hiệu quả của chúng.

Về mặt thực nghiệm: (1) Triển khai và kiểm chứng các thuật toán được đề xuất trên nhiều bộ dữ liệu thực tế có cấu trúc khác nhau (mạng xã hội, hệ thống cảm biến, dữ liệu hình ảnh, v.v.); (2) So sánh hiệu quả của các thuật toán với các phương pháp tốt nhất hiện nay, nhằm đánh giá giá trị thực tiễn và tính ổn định của lời giải thu được.

Về mặt học thuật: Các kết quả trong luận án đã được phản biện và công bố tại các hội nghị và tạp chí quốc tế có uy tín trong lĩnh vực tối ưu tổ hợp và học máy, góp phần khẳng định tính mới, giá trị khoa học và mức độ đóng góp của công trình nghiên cứu.

Các đóng góp nêu trên được trình bày theo hướng thận trọng, gắn với phạm vi các bài toán và các công trình liên quan được khảo sát. Cụ thể, luận án không chỉ phát biểu các bài toán nghiên cứu một cách thống nhất, mà còn phân tích rõ điều kiện áp dụng, bảo đảm xấp xỉ và độ phức tạp của từng thuật toán đề xuất. Nhờ đó, các kết quả đạt được có thể được đối chiếu trực tiếp với các nghiên cứu trước đây trong cùng lớp bài toán.

Tổng thể, các đóng góp của luận án không chỉ mở rộng phạm vi áp dụng của bài toán tối ưu hàm submodular, mà còn cung cấp các công cụ thực tiễn và lý thuyết cho việc giải quyết hiệu quả những bài toán tối ưu tổ hợp hiện đại.

5. Phương pháp nghiên cứu

Luận án sử dụng cách tiếp cận kết hợp giữa lý thuyết tối ưu tổ hợp và thực nghiệm trên dữ liệu thực để giải quyết các bài toán được đề xuất. Phương pháp nghiên cứu bao gồm ba trụ cột chính: thiết kế thuật toán, phân tích hiệu năng và đánh giá thực nghiệm.

Trên phương diện thiết kế thuật toán, luận án xây dựng các thuật toán xấp xỉ dựa trên các chiến lược tham lam cổ điển, tham lam nhanh và các kỹ thuật biến đổi bài toán nhằm khai thác cấu trúc đặc thù của hàm mục tiêu và ràng buộc.

Về mặt lý thuyết, mỗi thuật toán được đề xuất đều đi kèm với phân tích chặt chẽ về tỉ lệ xấp xỉ đạt được và độ phức tạp tính toán. Các phân tích được thực hiện trong khuôn khổ tổng quát nhằm đảm bảo khả năng mở rộng cho các ứng dụng đa dạng.

Trên phương diện thực nghiệm, các thuật toán được triển khai và kiểm thử trên nhiều bộ dữ liệu thực tế liên quan đến các bài toán ứng dụng như lan truyền ảnh hưởng trong mạng xã hội, bố trí cảm biến trong không gian giám sát và tối đa doanh thu. Các chỉ số đánh giá bao gồm: giá trị hàm mục tiêu thu được, độ phức tạp truy vấn, thời gian thực hiện, . . .

Cuối cùng, kết quả của các thuật toán đề xuất được so sánh với các thuật toán cơ sở mạnh nhất hiện có trong các nghiên cứu trước đây, cả về mặt định tính (mức độ áp dụng được vào các ứng dụng thực) lẫn định lượng (tỉ lệ hàm mục tiêu, thời gian thực thi), qua đó khẳng định giá trị đóng góp thực tiễn của các phương pháp nghiên cứu trong luận án.

6. Cấu trúc của luận án

Luận án được tổ chức thành bốn chương chính cùng phần kết luận. Thứ tự các chương được sắp xếp nhằm thể hiện mạch phát triển của nghiên cứu: từ kiến thức cơ sở, đến bài toán tối đa hàm submodular với ràng buộc chi phí như một bài toán nền tảng, tiếp theo là các mở rộng trên hai lớp hàm tổng quát hơn gồm k-submodular và DR-submodular, và cuối cùng là bài toán phủ submodular với chi phí tối thiểu theo hướng đối ngẫu. Cách sắp xếp này giúp làm rõ sự kế thừa từ bài toán tối đa submodular cổ điển, sang các biến thể mở rộng, rồi đến bài toán phủ đối ngẫu. Cấu trúc chi tiết như sau:

- **Chương 1** giới thiệu về bài toán tối ưu tổ hợp và các hàm submodular. Chương này trình bày các khái niệm nền tảng, bao gồm định nghĩa hàm submodular, các tính chất quan trọng và các kết quả lý thuyết kinh điển. Đây là phần chuẩn bị cần thiết để hiểu các bài toán và kỹ thuật được trình bày ở các chương sau.

- **Chương 2** tập trung vào Bài toán tối đa hàm submodular với ràng buộc chi phí - Bài toán nghiên cứu 1. Chương trình bày phát biểu bài toán, các khó khăn đặc thù của trường hợp tổng quát, thuật toán đề xuất cùng với phân tích lý thuyết và kiểm chứng thực nghiệm.

- **Chương 3** mở rộng nghiên cứu sang hai lớp hàm submodular tổng quát hơn: hàm k-submodular và DR-submodular. Cụ thể, chương này trình bày thuật toán cho bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm - Bài toán nghiên cứu 2, cũng như bài toán tối đa hàm DR-submodular với ràng buộc lực lượng - Bài toán nghiên cứu 3. Mỗi bài toán đều đi kèm với phân tích lý thuyết và thực nghiệm minh họa.

- **Chương 4** nghiên cứu Bài toán phủ submodular với chi phí tối thiểu - Bài toán nghiên cứu 4, tức bài toán MSC ở chương cuối của luận án. Chương này đề xuất giải thuật xấp xỉ cho bài toán phủ trong các trường hợp đơn điệu, đồng thời trình bày các đánh giá hiệu năng trên tập dữ liệu thực tế.

Cấu trúc luận án được thiết kế nhằm đảm bảo tính liền mạch, từ cơ sở lý thuyết đến ứng dụng cụ thể, giúp người đọc dễ dàng tiếp cận và đánh giá toàn diện quá trình nghiên cứu cũng như các đóng góp của luận án.

CHƯƠNG 1
GIỚI THIỆU VỀ TỐI ƯU HÀM SUBMODULAR

Chương này đóng vai trò đặt nền móng lý thuyết cho toàn bộ luận án. Các khái niệm, định nghĩa và công cụ toán học được trình bày trong chương là cơ sở quan trọng để hình thành mô hình và thiết kế thuật toán cho bốn bài toán nghiên cứu đã được xác định ở phần Lời mở đầu. Cụ thể, luận án tập trung vào các bài toán tối ưu tổ hợp có hàm mục tiêu thuộc lớp submodular, k-submodular và DR-submodular, đi kèm với các ràng buộc thực tế như chi phí, chi phí nhóm hoặc lực lượng. Do đó, việc hiểu rõ các thuộc tính cấu trúc của các lớp hàm này cũng như các kỹ thuật tối ưu xấp xỉ là tiền đề không thể thiếu.

Mặc dù mỗi bài toán nghiên cứu có đặc điểm riêng biệt về mặt hàm mục tiêu và ràng buộc, chúng đều có một điểm chung: đó là khai thác cấu trúc đặc biệt của hàm mục tiêu để xây dựng các thuật toán hiệu quả về mặt lý thuyết và thực nghiệm. Cụ thể, bài toán thứ nhất liên quan đến tối đa hàm submodular không đơn điệu với ràng buộc chi phí; bài toán thứ hai là tối ưu hàm k-submodular với ràng buộc chi phí nhóm; bài toán thứ ba là tối ưu hàm DR-submodular không đơn điệu với ràng buộc lực lượng; bài toán thứ tư là phủ submodular với mục tiêu tối thiểu chi phí. Các hàm này yêu cầu cách tiếp cận riêng biệt, tuy nhiên đều xuất phát từ cùng một nền tảng toán học về tối ưu submodular.

Đầu tiên, luận án trình bày các khái niệm cơ bản trong tối ưu tổ hợp, đặc biệt là thuật toán xấp xỉ và các kỹ thuật đánh giá hiệu quả thuật toán. Sau đó, luận án đi sâu vào định nghĩa và thuộc tính của hàm submodular, tiếp theo là mở rộng sang các lớp hàm k-submodular và DR-submodular. Ngoài ra, chương còn giới thiệu một số ràng buộc phổ biến như ràng buộc chi phí, ràng buộc nhóm và ràng buộc lực lượng – vốn là các loại ràng buộc quan trọng trong mô hình hóa bài toán thực tế. Cuối cùng là phần trình bày về các ứng dụng của bài toán tối ưu hàm submodular. Vì vậy, chương này không chỉ trình bày kiến thức lý thuyết, mà còn xây dựng nền tảng toán học chung cho các bài toán nghiên cứu của luận án.

1.1. Bài toán tối ưu tổ hợp và các thuật toán xấp xỉ

Bài toán tối ưu hàm submodular là một trường hợp đặc biệt trong lớp các bài toán tối ưu tổ hợp. Do đó, việc hiểu rõ cấu trúc và tính chất của các bài toán tối ưu tổ hợp là cần thiết để xây dựng các thuật toán hiệu quả cho các bài toán phức tạp hơn như tối ưu hàm submodular có ràng buộc. Vì vậy, luận án trước hết sẽ tập trung trình bày những khái niệm và đặc điểm nền tảng của tối ưu tổ hợp nhằm xây dựng cơ sở lý thuyết cho các phần sau.

1.1.1. Bài toán tối ưu tổ hợp

Tối ưu tổ hợp (*combinatorial optimization*) là một nhánh trọng yếu trong lĩnh vực tối ưu rời rạc, đóng vai trò nền tảng trong nhiều bài toán thực tiễn như thiết kế mạng, lập lịch, phân bổ tài nguyên, học máy. Một bài toán tối ưu tổ hợp được phát biểu cụ thể như sau:

Định nghĩa 1.1 (Bài toán tối ưu tổ hợp [62])**.** Cho một tập hợp hữu hạn Ω gọi là tập nghiệm khả thi (*hoặc trạng thái chấp nhận được*) và một hàm mục tiêu $f : \Omega \to \mathbb{R}$. Mục tiêu của bài toán tối ưu tổ hợp là: $s^* = \arg\max(\min)_{s\in\Omega} f(s)$, tùy theo bài toán là cực đại hay cực tiểu.

Không gian nghiệm Ω trong bài toán tối ưu tổ hợp thường mang bản chất rời rạc và có thể được biểu diễn dưới nhiều dạng hình thức khác nhau, ví dụ:

- Tập hợp con: $\Omega = 2^V$ với V là tập hữu hạn. Dạng này phổ biến trong các bài toán như tập phủ, tối ưu hàm submodular.
- Vector nhị phân: $\Omega = \{0, 1\}^n$. Mỗi nghiệm được biểu diễn bởi một chuỗi nhị phân thể hiện lựa chọn/bỏ qua một phần tử, thường gặp trong bài toán ba lô.
- Vector nguyên: $\Omega \subseteq \mathbb{Z}^n$. Mỗi nghiệm là một vector với các biến nguyên, dùng trong các bài toán lập lịch, phân bổ rời rạc hoặc bài toán tối ưu hàm DR-submodular.
- Miền rời rạc nhiều giá trị: $\Omega = \{0, 1, \ldots, k\}^n$, đặc trưng cho các bài toán phân cụm rời rạc hoặc bài toán tối đa hàm k-submodular.

Bài toán tối ưu tổ hợp có nhiều đặc điểm nổi bật về mặt lý thuyết và tính toán:

- Không gian nghiệm lớn: số lượng nghiệm khả thi thường tăng theo cấp số mũ theo số biến n, ví dụ như 2^n đối với tập hợp con.
- Cấu trúc rời rạc và tổ hợp: nghiệm không nằm trong không gian liên tục, không thể sử dụng các công cụ giải tích như đạo hàm hay gradient, mà phải dùng các chiến lược liệt kê, nhánh cận, hoặc heuristic.
- Tính toán khó khăn: phần lớn các bài toán tối ưu tổ hợp là NP-Khó (***Định nghĩa 1.5***), tức là không có thuật toán chính xác chạy trong thời gian đa thức theo kích thước dữ liệu đầu vào trừ khi $P = NP$.

Để minh họa rõ hơn cho khái niệm bài toán tối ưu tổ hợp, ta xét một số bài toán kinh điển sau:

- **Bài toán Tập phủ đỉnh (Vertex Cover)** [39]: Cho một đồ thị vô hướng $G = (V, E)$, mục tiêu là tìm một tập con các đỉnh $S \subseteq V$ sao cho mỗi cạnh $e \in E$ đều có ít nhất một đỉnh thuộc S, đồng thời kích thước của tập $|S|$ là nhỏ nhất. Không gian trạng thái của bài toán là tập tất cả các tập con của tập đỉnh V, tức $\Omega = \{S \subseteq V\}$, với kích thước không gian là $2^{|V|}$. Trong đó, không gian nghiệm khả thi $\mathcal{F} \subseteq \Omega$ bao gồm các tập con thỏa mãn ràng buộc: với mọi $(u, v) \in E,\ u \in S \vee v \in S$. Bài toán có thể được phát biểu như một bài toán tối ưu tổ hợp: $S^* \in \arg\min_{S\in\mathcal{F}} |S|$. Mặc dù đây là một bài toán NP-đầy đủ,

nhưng vẫn tồn tại một thuật toán xấp xỉ với tỉ lệ xấp xỉ $1/2$, tức là luôn tìm được tập phủ có kích thước không vượt quá hai lần so với tập phủ tối ưu. Bài toán này có nhiều ứng dụng quan trọng trong các lĩnh vực như tối ưu hóa mạng, lập lịch, phân tích xã hội và thiết kế hệ thống phân tán.

- **Bài toán Ba lô (Knapsack Problem)** [89]: Cho n vật phẩm, trong đó vật phẩm thứ i có trọng lượng $a_i \in \mathbb{R}_+$ và giá trị $c_i \in \mathbb{R}_+$, cùng một ba lô có sức chứa tối đa $B \in \mathbb{R}_+$. Mục tiêu là chọn một tập con các vật phẩm sao cho tổng giá trị là lớn nhất, đồng thời tổng trọng lượng không vượt quá dung lượng ba lô. Bài toán có thể được phát biểu dưới dạng bài toán tối ưu tổ hợp nhị phân như sau:

$$\max_{x\in\{0,1\}^n} \sum_{i=1}^{n} c_i x_i, \quad \text{s.t.} \quad \sum_{i=1}^{n} a_i x_i \leq B.$$

Trong đó, biến nhị phân $x_i \in \{0, 1\}$ biểu thị việc chọn (1) hoặc không chọn (0) vật phẩm thứ i. Không gian trạng thái của bài toán là tập $\Omega = \{0, 1\}^n$, gồm 2^n cấu hình lựa chọn các vật phẩm. Không gian nghiệm khả thi là tập con $\mathcal{F} \subseteq \Omega$ thỏa mãn ràng buộc tổng trọng lượng không vượt quá B. Đây là một bài toán NP-đầy đủ, do đó khó có thể tìm nghiệm tối ưu trong thời gian đa thức, trừ khi $P = NP$. Tuy nhiên, bài toán có thể được giải bằng quy hoạch động trong thời gian giả đa thức $O(nB)$, hoặc xấp xỉ bằng các thuật toán tham lam dựa trên mật độ giá trị – trọng lượng c_i/a_i, đặc biệt hiệu quả trong trường hợp bài toán cho phép chia nhỏ vật phẩm.

Từ những cơ sở và ví dụ nêu trên, ta thấy bài toán tối ưu tổ hợp không chỉ phong phú về hình thức biểu diễn mà còn phức tạp về mặt tính toán. Những đặc điểm này sẽ là nền tảng để nghiên cứu sâu hơn các lớp bài toán có cấu trúc đặc biệt như tối ưu submodular trong các phần tiếp theo của luận án.

1.1.2. Độ phức tạp tính toán và phân lớp bài toán

Độ phức tạp tính toán là một khái niệm trung tâm trong khoa học máy tính lý thuyết, cung cấp khung hình thức để đánh giá lượng tài nguyên (ví dụ thời gian hoặc bộ nhớ) cần thiết nhằm giải quyết một bài toán. Trong bối cảnh tối ưu tổ hợp, rất nhiều bài toán cổ điển đã biết có thuật toán thời gian đa thức, nhưng cũng có nhiều bài toán quan trọng mà cho đến nay vẫn chưa biết có thuật toán đa thức hay không. Điều này dẫn tới nhu cầu phân lớp các bài toán theo độ phức tạp tính toán của chúng. Để dễ dàng phân lớp bài toán, việc chuẩn hoá các bài toán về dạng bài toán quyết định đóng vai trò quan trọng. Trong đó, bài toán quyết định được định nghĩa như sau:

Định nghĩa 1.2 (Bài toán quyết định [69])**.** Một bài toán quyết định là một cặp $\mathcal{P} = (X, Y)$, trong đó $X \subseteq \{0, 1\}^*$ là một ngôn ngữ có thể quyết định được trong thời gian đa thức và $Y \subseteq X$. Các phần tử của X được gọi là các instance của $\mathcal{P}$; các phần tử của Y được gọi là *yes–instances*, còn các phần tử của $X \setminus Y$ được gọi là *no–instances*.

Bài toán quyết định (X, Y) có thể được xem như một bài toán tính toán với tập giá trị $(X, \{(x, 1) : x \in Y\} \cup \{(x, 0) : x \in X \setminus Y\})$. Do đó, một thuật toán cho bài toán quyết định (X, Y) chính là một thuật toán tính hàm $f : X \to \{0, 1\}$, được xác định bởi $f(x) = 1$ nếu $x \in Y$ và $f(x) = 0$ nếu $x \in X \setminus Y$.

Ở đây $X \subseteq \{0, 1\}^*$ được hiểu là một ngôn ngữ trên bảng chữ cái nhị phân, tức là một tập con bất kỳ của tập tất cả các xâu nhị phân. Mệnh đề "X có thể quyết định được trong thời gian đa thức" có nghĩa là tồn tại một thuật toán (hay tương đương, một máy Turing) sao cho với mọi xâu nhị phân x, thuật toán luôn dừng sau thời gian bị chặn bởi một đa thức theo độ dài $|x|$ và trả về kết quả đúng của mệnh đề "$x \in X$". Nói cách khác, việc kiểm tra một xâu x có thuộc X hay không có thể được thực hiện trong thời gian đa thức theo kích thước dữ liệu vào. Khi đó lớp bài toán P và NP được định nghĩa như sau:

Định nghĩa 1.3 (Lớp bài toán P [69])**.** Lớp P là lớp tất cả các bài toán quyết định mà tồn tại một thuật toán thời gian đa thức giải bài toán đó. Nói cách khác, một phần tử của P là một cặp (X, Y) với $Y \subseteq X \subseteq \{0, 1\}^*$, trong đó cả X và Y đều là các ngôn ngữ có thể quyết định được trong thời gian đa thức.

Định nghĩa 1.4 (Lớp bài toán NP [69])**.** Một bài toán quyết định $\mathcal{P} = (X, Y)$ được gọi là thuộc lớp NP nếu tồn tại một đa thức p và một bài toán quyết định $\mathcal{P}' = (X', Y')$ thuộc lớp P sao cho $X' := \{x\#c | x \in X, c \in \{0, 1\}^{[p(\text{size}(x))]}\}$ và $Y = \{y \in X | \exists c \in \{0, 1\}^{[p(\text{size}(y))]}$ với $y\#c \in Y'\}$. Ở đây $x\#c$ ký hiệu phép nối chuỗi x, ký hiệu đặc biệt # và chuỗi c. Một chuỗi c sao cho $y\#c \in Y'$ được gọi là một chứng nhận (certificate) cho y và một thuật toán cho $\mathcal{P}'$ được gọi là thuật toán kiểm tra (certificate–checking algorithm).

Từ Định nghĩa 1.4 suy ra ngay rằng $P \subseteq NP$ [69]. Ngoài ra, Mệnh đề 15.14 trong [69] cho thấy định nghĩa lớp NP thông qua chứng nhận đa thức và thuật toán kiểm tra là tương đương với định nghĩa kinh điển dựa trên máy Turing không đơn định: một bài toán quyết định thuộc lớp NP khi và chỉ khi tồn tại một thuật toán không đơn định chạy trong thời gian đa thức giải bài toán đó. Điều này củng cố cách hiểu trực giác rằng NP là lớp các bài toán mà, dù ta chưa biết cách tìm nghiệm trong thời gian đa thức, nhưng có thể kiểm tra một chứng nhận của nghiệm trong thời gian đa thức.

Định nghĩa 1.5 (Bài toán NP-khó [69])**.** Một bài toán tính toán $\mathcal{P}$ được gọi là NP-khó (NP-hard) nếu mọi bài toán trong lớp NP đều giảm đa thức (polynomially reduce) về $\mathcal{P}$.

Trong đó, ta nói rằng bài toán tính toán $\mathcal{P}_1$ giảm đa thức về bài toán tính toán $\mathcal{P}_2$ nếu tồn tại một thuật toán thời gian đa thức có oracle cho $\mathcal{P}_1$ sử dụng $\mathcal{P}_2$ làm oracle, tức là thuật toán này được phép trong quá trình tính toán gọi tới một "hộp đen" giải đúng mọi instance của $\mathcal{P}_2$ và tổng thời gian (bao gồm cả số lần gọi oracle) bị chặn bởi một đa thức theo kích thước đầu vào của $\mathcal{P}_1$ [69].

Sự hiện diện của độ phức tạp NP-khó đồng nghĩa với việc không thể kỳ vọng thiết kế được thuật toán chính xác có thời gian chạy đa thức cho các bài toán này (trừ khi giả thuyết $P = NP$ đúng). Do đó, thay vì tìm lời giải chính xác, hướng tiếp cận khả thi hơn là phát triển các thuật toán xấp xỉ, trong đó ta chấp nhận một mức độ sai lệch có thể kiểm soát được giữa nghiệm tìm được và nghiệm tối ưu. Đây chính là động lực thúc đẩy nghiên cứu các thuật toán xấp xỉ, sẽ được trình bày trong các phần tiếp theo của chương này.

1.1.3. Thuật toán xấp xỉ

Trong bối cảnh các bài toán tối ưu rời rạc thường có độ phức tạp rất cao (nhiều bài toán là NP-khó hoặc NP-đầy đủ), việc tìm nghiệm tối ưu bằng các thuật toán chính xác có thể đòi hỏi thời gian và bộ nhớ không khả thi trong thực tế. Khi đó, thuật toán heuristic đóng vai trò thiết yếu: chúng được phát triển như các kỹ thuật xấp xỉ (approximate techniques) nhằm tìm các nghiệm xấp xỉ hoặc thậm chí nghiệm một phần cho bài toán tối ưu, với độ phức tạp thời gian và không gian ở mức chấp nhận được. Theo Kokash [68], có thể hiểu rộng heuristic là những thuật toán cho gần đúng lời giải chính xác hoặc chỉ cho lời giải trên một phần các thể hiện của bài toán, thường dựa trên các cơ chế mang tính kinh nghiệm, các "quy tắc ngón tay cái" hoặc việc khai thác cấu trúc cụ thể của bài toán hơn là trên các bảo đảm lý thuyết chặt chẽ; hiệu quả của chúng về cơ bản được đánh giá thông qua các nghiên cứu thực nghiệm trên những bộ dữ liệu và kịch bản cụ thể và nói chung không đi kèm một bảo đảm tổng quát về khoảng cách tới nghiệm tối ưu trong trường hợp xấu nhất cho mọi thể hiện.

Trái lại, lý thuyết thuật toán xấp xỉ (approximation algorithms) tiếp cận cùng lớp bài toán từ một góc nhìn hình thức hơn: Korte [69] định nghĩa một thuật toán α-xấp xỉ là một thuật toán đa thức luôn trả về nghiệm có giá trị nằm trong một nhân tử α so với giá trị tối ưu trên mọi thể hiện của bài toán, trong đó α đóng vai trò là "performance guarantee" hay "approximation ratio". Cách tiếp cận này chấp nhận nới lỏng yêu cầu tối ưu tuyệt đối, nhưng đồng thời định lượng một cách rõ ràng mức "tổn thất" so với tối ưu trong trường hợp xấu nhất, qua đó cung cấp một khung toán học chặt chẽ để nghiên cứu các thuật toán cho nghiệm gần tối ưu trong thời gian đa thức và làm nổi bật sự khác biệt về mặt bảo đảm lý thuyết giữa các thuật toán xấp xỉ và các thuật toán heuristic thuần túy. Cụ thể định nghĩa thuật toán xấp xỉ được phát biểu như sau:

Định nghĩa 1.6 (Thuật toán xấp xỉ [69])**.** Xét một bài toán tối ưu tổ hợp có tập nghiệm khả thi Ω và hàm mục tiêu $f : \Omega \to \mathbb{R}$. Gọi $x^* \in \arg\max_{x\in\Omega} f(x)$ (hoặc $\arg\min$ trong trường hợp cực tiểu) là nghiệm tối ưu. Một thuật toán đa thức $\mathcal{A}$ được gọi là thuật toán xấp xỉ với tỉ lệ xấp xỉ α nếu với mọi đầu vào (input), nghiệm $\hat{x} = \mathcal{A}(\text{input})$ do thuật toán

trả về thoả mãn:

$$\begin{cases} f(\hat{x}) \geq \alpha \cdot f(x^*) & \text{nếu là bài toán cực đại, với } 0 < \alpha \leq 1, \\ f(\hat{x}) \leq \alpha \cdot f(x^*) & \text{nếu là bài toán cực tiểu, với } \alpha \geq 1. \end{cases}$$

Trong đó, thuật toán đa thức là thuật toán có thời gian chạy trong trường hợp tệ nhất bị chặn bởi một đa thức theo kích thước đầu vào n, tức $T(n) = O(n^k)$ với một hằng số k. Điều kiện thuật toán xấp xỉ là thuật toán đa thức là thiết yếu: nếu bỏ qua ràng buộc này thì ta có thể dùng thuật toán chính xác (thường có thời gian chạy luỹ thừa) với tỉ lệ xấp xỉ $\alpha = 1.0$, khiến khái niệm thuật toán xấp xỉ trở nên tầm thường. Hệ số α ở đây gọi là tỉ lệ xấp xỉ (approximation ratio), dùng để đo lường chất lượng tương đối giữa nghiệm xấp xỉ và nghiệm tối ưu. Giá trị α càng gần 1 thì nghiệm xấp xỉ càng gần nghiệm tối ưu. Trong một số trường hợp, α có thể là một hằng số, hoặc phụ thuộc vào kích thước đầu vào, chẳng hạn $\alpha(n) = \ln n$. Ngoài ra, cũng có thể sử dụng dạng biểu diễn là $\alpha f(\hat{x}) \geq f(x^*)$ đối với bài toán cực đại (hoặc $\alpha f(\hat{x}) \leq f(x^*)$ đối với bài toán cực tiểu), khi đó hệ số xấp xỉ α lớn hơn 1 và càng nhỏ thì nghiệm xấp xỉ càng tốt đối với bài toán cực đại và ngược lại đối với bài toán cực tiểu. Trong luận án này, nghiên cứu sinh đồng nhất sử dụng hệ số xấp xỉ $\alpha \in (0, 1]$ cho bài toán cực đại và $\alpha \geq 1$ cho bài toán cực tiểu.

Trong một số bài toán tối ưu tổ hợp, đặc biệt là các bài toán cực tiểu có ràng buộc như phủ submodular tối thiểu chi phí (submodular cover), việc thỏa mãn nghiêm ngặt các ràng buộc có thể khiến việc tối ưu trở nên quá khó, hoặc không thể đạt được hiệu suất tốt. Khi đó, một hướng tiếp cận mềm dẻo hơn là sử dụng thuật toán xấp xỉ hai tiêu chí (bicriteria approximation), cho phép vi phạm nhẹ ràng buộc trong khi vẫn đảm bảo một giới hạn gần tối ưu cho hàm mục tiêu.

Định nghĩa 1.7 (Thuật toán xấp xỉ hai tiêu chí [64])**.** Giả sử một bài toán tối ưu cực tiểu được phát biểu dưới dạng:

$$x^* \leftarrow \underset{x \in \Omega : f(x) \geq T}{\operatorname{argmin}} \; c(x),$$

trong đó $f : \Omega \to \mathbb{R}_+$ là hàm tập hợp, $c : \Omega \to \mathbb{R}_+$ là hàm chi phí và $T > 0$ là ngưỡng yêu cầu. Một thuật toán đa thức $\mathcal{A}$ được gọi là thuật toán xấp xỉ tiêu chí kép với cặp hệ số (α, β) nếu với mọi đầu vào, nghiệm $\hat{x} = \mathcal{A}(\text{input})$ thỏa mãn:

$$f(\hat{x}) \geq \beta \cdot T, \quad c(\hat{x}) \leq \alpha \cdot c(x^*),$$

trong đó x^* là nghiệm tối ưu của bài toán gốc.

Tại đây, hệ số $\alpha \geq 1$ đo lường mức độ lệch về chi phí so với nghiệm tối ưu, trong khi hệ số $0 \leq \beta \leq 1$ cho phép vi phạm giới hạn ràng buộc về hàm mục tiêu. Ngoài ra, cũng có thể sử dụng dạng biểu diễn là $\alpha c(\hat{x}) \leq c(x^*)$ khi đó hệ số xấp xỉ α nằm trong

khoảng $(0, 1)$. Đặc biệt, nếu $\beta = 1$, thuật toán trở thành xấp xỉ đơn tiêu chí thông thường. Tuy nhiên, trong nhiều bài toán như phủ submodular, việc chấp nhận $\beta \leq 1$ là cần thiết để đạt hiệu suất tốt về chi phí. Định nghĩa này cung cấp cơ sở để phân tích các thuật toán cho nhóm bài toán phủ có ngưỡng yêu cầu, trong đó chất lượng nghiệm cần được đánh giá đồng thời theo chi phí và mức độ thỏa mãn hàm submodular.

Một số điểm quan trọng cần làm rõ là sự khác biệt giữa ba loại thuật toán:

- Thuật toán chính xác luôn tìm được nghiệm tối ưu sau hữu hạn bước; tuy nhiên với các bài toán NP-khó thì chi phí tính toán có thể là hàm mũ.

- Thuật toán heuristic được thiết kế để tìm nghiệm tốt trong thực tế, nhưng không có bảo đảm lý thuyết nào về chất lượng nghiệm và không đảm bảo được với mọi đầu vào của bài toán.

- Thuật toán xấp xỉ cân bằng giữa hiệu quả tính toán và chất lượng nghiệm, với cam kết lý thuyết thông qua hệ số xấp xỉ α.

Một ví dụ điển hình là bài toán tập phủ (Set Cover), trong đó ta cần chọn tập hợp con nhỏ nhất sao cho bao phủ toàn bộ tập phần tử. Một thuật toán tham lam đơn giản, tại mỗi bước chọn tập con phủ được nhiều phần tử chưa phủ nhất, cho phép đạt tỉ lệ xấp xỉ $\alpha(n) = \ln(n)$ – và đây là chặn tốt nhất có thể đạt được trong điều kiện giả thuyết $P \neq NP$ [32]. Trường hợp này minh hoạ rõ ràng cho hiệu quả thực tiễn và bảo đảm lý thuyết mà thuật toán xấp xỉ mang lại trong việc giải quyết các bài toán tổ hợp NP-khó.

Với những đặc điểm nêu trên, thuật toán xấp xỉ đóng vai trò then chốt trong tối ưu tổ hợp hiện đại, đặc biệt trong các bài toán có không gian nghiệm lớn và ràng buộc phức tạp. Các kỹ thuật thiết kế thuật toán xấp xỉ sẽ tiếp tục được trình bày trong phần sau của chương này.

1.1.4. Đánh giá và bảo đảm lý thuyết

Đánh giá lý thuyết là một thành phần cốt lõi trong phân tích thuật toán, đặc biệt trong bối cảnh các bài toán tối ưu tổ hợp mà thuật toán chính xác là không khả thi. Khi đó, một trong những mục tiêu chính là thiết lập các bảo đảm lý thuyết (theoretical guarantees) về hiệu suất của thuật toán xấp xỉ. Các bảo đảm này không chỉ cung cấp thông tin về chất lượng tương đối giữa nghiệm tìm được và nghiệm tối ưu, mà còn giúp so sánh các thuật toán khác nhau một cách có cơ sở.

Phổ biến nhất là phân tích hiệu suất thuật toán thông qua tỉ lệ xấp xỉ đã định nghĩa ở mục trước. Tuy nhiên, bên cạnh việc xác định một hệ số xấp xỉ cụ thể, một câu hỏi quan trọng là liệu hệ số xấp xỉ tốt nhất đó có thể tiếp tục được cải thiện hay không. Khi một hệ số xấp xỉ không thể vượt qua một ngưỡng nhất định (dưới giả thuyết $P \neq NP$), ta nói bài toán có độ khó trong việc xấp xỉ (hardness of approximation).

Định nghĩa 1.8 (Độ khó trong việc xấp xỉ [32])**.** Một bài toán tối ưu được gọi là khó xấp xỉ tới hệ số β nếu không tồn tại thuật toán đa thức nào có thể đạt hệ số xấp xỉ tốt hơn β, trừ khi $P = NP$.

Kết quả này thường được chứng minh thông qua các kỹ thuật giảm GAP (gap-preserving reductions) hoặc lý thuyết chứng minh xác suất (probabilistically checkable proofs – PCP). Những kỹ thuật này cho phép chuyển độ khó từ các bài toán quyết định NP-đầy đủ sang các bài toán tối ưu, đồng thời bảo tồn khoảng cách giữa nghiệm tốt và nghiệm xấu, từ đó thiết lập rào cản cho thuật toán xấp xỉ.

Một khái niệm quan trọng khác là xấp xỉ chặt, tức là thuật toán xấp xỉ đã đạt hiệu suất tối ưu về mặt lý thuyết.

Định nghĩa 1.9 (Xấp xỉ chặt [32])**.** Gọi α là tỉ lệ xấp xỉ đạt được bởi một thuật toán xấp xỉ và β là hệ số xấp xỉ tốt nhất có thể đạt được đối với bài toán đó. Nếu $\alpha = \beta$, ta nói hệ số xấp xỉ là chặt (tight) và thuật toán đạt hiệu suất tối ưu về mặt lý thuyết.

Một ví dụ tiêu biểu minh hoạ cho khái niệm này là bài toán tối đa hàm submodular đơn điệu với ràng buộc lực lượng. Thuật toán tham lam cổ điển cho bài toán này đạt tỉ lệ xấp xỉ $1 - 1/e$ và đây là một hệ số chặt, tức không thể cải thiện trừ khi $P = NP$ [93]. Cụ thể, nếu $f : 2^V \to \mathbb{R}_+$ là một hàm submodular đơn điệu và k là ràng buộc về kích thước, bài toán: $\max\{f(S) : S \subseteq V, |S| \leq k\}$ được giải bằng thuật toán tham lam với bảo đảm:

$$f(S_{\text{Greedy}}) \geq (1 - \frac{1}{e}) f(S^*),$$

trong đó S_{Greedy} là nghiệm của thuật toán tham lam, S^* là nghiệm tối ưu. Đây là tỉ lệ tốt nhất có thể đạt được trong thời gian đa thức theo kích thước dữ liệu đầu vào.

Do đó, việc đánh giá hiệu suất thuật toán không chỉ dừng lại ở mức độ phân tích tỉ lệ xấp xỉ đạt được, mà còn cần đối chiếu với giới hạn lý thuyết để xác định mức độ tối ưu của chính bản thân chiến lược thiết kế thuật toán. Điều này cung cấp cơ sở vững chắc để so sánh giữa các phương pháp và là động lực định hướng nghiên cứu các thuật toán tốt hơn trong các trường hợp bài toán chưa đạt hệ số xấp xỉ tốt nhất. Trong phạm vi luận án, việc đánh giá thuật toán được xem xét theo một số tiêu chí chính: tỉ lệ xấp xỉ hoặc cặp hệ số xấp xỉ, độ phức tạp thời gian, số lần truy vấn oracle đến hàm mục tiêu, yêu cầu bộ nhớ, số vòng xử lý trong mô hình luồng hoặc song song, và chất lượng thực nghiệm trên các bộ dữ liệu được sử dụng. Các tiêu chí này giúp kết nối giữa bảo đảm lý thuyết và khả năng triển khai của thuật toán trong các bài toán cụ thể.

1.1.5. Các phương pháp thiết kế thuật toán xấp xỉ

Việc thiết kế các thuật toán xấp xỉ hiệu quả cho các bài toán tối ưu tổ hợp NP-khó đòi hỏi sự kết hợp giữa tư duy thuật toán và hiểu biết sâu sắc về cấu trúc của bài toán. Các phương pháp thiết kế không chỉ cung cấp khuôn mẫu xây dựng thuật toán, mà còn

quyết định trực tiếp đến chất lượng nghiệm, độ phức tạp truy vấn, khả năng mở rộng và tính thích ứng với môi trường tính toán cụ thể. Trong phần này, nghiên cứu sinh trình bày các phương pháp phổ biến có liên quan trực tiếp đến các thuật toán được sử dụng trong luận án, bao gồm: tham lam, ngẫu nhiên, song song và luồng.

a) Phương pháp tham lam

Phương pháp tham lam (greedy) là một khuôn mẫu tổng quát để thiết kế thuật toán cho các bài toán tối ưu. Về trực giác, một thuật toán tham lam tiến hành qua một dãy các bước; ở mỗi bước nó đưa ra lựa chọn "trông có vẻ tốt nhất ở thời điểm hiện tại", tức là lựa chọn cục bộ tối ưu với hi vọng chuỗi các lựa chọn cục bộ đó dẫn tới nghiệm tối ưu toàn cục. Trong nhiều bài toán tối ưu tổ hợp, việc áp dụng các thuật toán chính xác quá tốn kém về tài nguyên; khi đó, một thuật toán tham lam thường cho lời giải đơn giản và hiệu quả hơn cả về thời gian lẫn cài đặt, mặc dù không phải lúc nào cũng cho nghiệm tối ưu.

Về mặt khái niệm, có thể xem phương pháp tham lam như sau. Ta xét một bài toán tối ưu mà nghiệm là các đối tượng trong một không gian cấu trúc. Thuật toán duy trì một nghiệm bộ phận S (ban đầu thường là rỗng) và lặp lại quá trình: (i) xác định tập ứng viên khả thi C có thể "mở rộng" nghiệm hiện tại mà vẫn bảo toàn tính hợp lệ, (ii) chọn phần tử $c \in C$ tốt nhất theo một tiêu chí cục bộ (tiêu chí tham lam), rồi gộp c vào nghiệm. Các quyết định này là không hồi lại: một khi đã chọn thì không bị thay đổi về sau. Việc thiết kế một thuật toán tham lam thường tuân theo ba bước khái quát. Thứ nhất, ta biểu diễn bài toán tối ưu thành dạng mà tại mỗi bước ta đưa ra một lựa chọn và còn lại một bài toán con duy nhất cần giải. Thứ hai, ta chứng minh tồn tại một nghiệm tối ưu luôn thực hiện được lựa chọn tham lam ở bước đầu tiên (thuộc tính lựa chọn tham lam - greedy-choice property). Thứ ba, ta chứng minh cấu trúc con tối ưu (optimal substructure): sau khi đã cố định lựa chọn tham lam, nếu bài toán con còn lại được giải tối ưu và ghép lại với lựa chọn đó thì thu được nghiệm tối ưu cho bài toán ban đầu. Khi hai điều kiện này được bảo đảm, lược đồ tham lam ở trên sẽ dẫn tới nghiệm tối ưu cho bài toán cần xét.

Một nét khác biệt quan trọng giữa phương pháp tham lam và quy hoạch động nằm ở trật tự ra quyết định. Trong quy hoạch động, ta giải các bài toán con trước (tiếp cận "từ dưới lên"), rồi dựa trên các giá trị tối ưu của bài toán con để đưa ra lựa chọn ở bài toán lớn. Ngược lại, trong phương pháp tham lam, thuật toán đưa ra lựa chọn ngay từ đầu, dựa trên thông tin cục bộ của bài toán hiện tại, rồi mới giải bài toán con còn lại (tiếp cận "từ trên xuống"). Điều này cho phép xây dựng các thuật toán tham lam đơn giản và trong nhiều trường hợp có độ phức tạp thấp hơn. Tuy nhiên, việc chứng minh tính tối ưu của thuật toán tham lam thường đòi hỏi các lập luận tinh tế, và không phải bài toán nào có cấu trúc con tối ưu cũng thỏa mãn thuộc tính lựa chọn tham lam. Mặt

khác, với quy hoạch động, việc thiết kế trạng thái, quan hệ truy hồi và tối ưu độ phức tạp cũng là một thách thức quan trọng trong nhiều bài toán.

Phương pháp tham lam xuất hiện trong nhiều thuật toán kinh điển. Chẳng hạn, bài toán chọn tập hoạt động có kích thước lớn nhất mà không giao nhau (activity-selection) có thể giải tối ưu bằng cách luôn chọn hoạt động kết thúc sớm nhất trong số các hoạt động còn lại; thuật toán xây dựng mã Huffman cho bài toán nén dữ liệu nhị phân cũng là một ví dụ điển hình của việc lặp lại bước "gộp hai nút có tần suất nhỏ nhất" để thu được một cây mã tối ưu. Các thuật toán cây khung nhỏ nhất (như Kruskal, Prim), thuật toán Dijkstra cho đường đi ngắn nhất với trọng số không âm và heuristic tập phủ dựa trên Chvátal đều có thể được nhìn nhận như những hiện thân cụ thể của khuôn mẫu tham lam.

Mặt khác, không phải mọi bài toán tối ưu đều có thể giải bằng một chiến lược tham lam đơn giản. Cặp bài toán ba lô phân số và ba lô 0-1 là một ví dụ minh họa rõ nét cho sự khác biệt giữa chiến lược tham lam và quy hoạch động. Với bài toán ba lô phân số, do cho phép lấy một phần vật phẩm, việc sắp xếp vật phẩm theo tỉ số giá trị/trọng lượng giảm dần rồi lần lượt chọn vào ba lô cho ta nghiệm tối ưu. Ngược lại, trong bài toán ba lô 0-1, mỗi vật phẩm chỉ có thể được chọn hoặc không chọn, nên chiến lược tham lam theo cùng tỉ số này có thể thất bại. Bài toán ba lô 0-1 thường được giải bằng quy hoạch động nhờ khai thác cấu trúc con tối ưu và các bài toán con chồng lấp.

b) Phương pháp ngẫu nhiên.

Phương pháp ngẫu nhiên đóng vai trò trung tâm trong thiết kế các thuật toán xấp xỉ hiện đại cho bài toán tối ưu tổ hợp, đặc biệt khi các chiến lược tất định tỏ ra thiếu linh hoạt hoặc không đạt hiệu suất mong muốn. Ngẫu nhiên hóa cho phép thuật toán khai thác tính đa dạng trong không gian nghiệm, tránh rơi vào những cấu hình xấu và đạt được nghiệm tốt hơn về mặt trung bình hoặc với xác suất cao. Cách tiếp cận này đặc biệt hiệu quả trong các tình huống không thể thiết kế được chiến lược lựa chọn cố định tối ưu, hoặc không có cấu trúc thuận lợi để sử dụng các kỹ thuật tổ hợp truyền thống.

Trong thiết lập ngẫu nhiên, hiệu suất của thuật toán thường được đánh giá theo hai hướng chính: (i) hiệu suất kỳ vọng, tức là nghiệm trả về có giá trị kỳ vọng gần nghiệm tối ưu và (ii) hiệu suất với xác suất cao, tức là nghiệm thu được có chất lượng gần tối ưu với xác suất ít nhất $1 - \delta$, trong đó δ là một hằng số nhỏ. Cả hai thiết lập đều cung cấp sự bảo đảm lý thuyết rõ ràng về chất lượng nghiệm, giúp thuật toán ngẫu nhiên có vị thế ngang hàng, hoặc đôi khi vượt trội, so với các phương pháp tất định trong nhiều bối cảnh.

Một trong những kỹ thuật ngẫu nhiên hóa điển hình là làm tròn xác suất, được áp dụng rộng rãi trong các bài toán như bài tập tập phủ, bài toán thỏa mãn tối đa (MAX-SAT) và bài toán vị trí cơ sở (Facility Location). Trong kỹ thuật này, bài toán

rời rạc ban đầu được nới lỏng về bài toán liên tục (thường là quy hoạch tuyến tính) và nghiệm liên tục thu được được làm tròn về nghiệm nhị phân thông qua quá trình ngẫu nhiên hóa theo phân phối tỉ lệ. Phân tích hiệu suất của kỹ thuật này thường dựa trên các bất đẳng thức xác suất cổ điển như bất đẳng thức Markov, Chernoff hoặc Hoeffding, nhằm đảm bảo tổng sai lệch do ngẫu nhiên không vượt quá giới hạn cho phép với xác suất cao.

Ngoài làm tròn, ngẫu nhiên hóa còn được sử dụng trong các chiến lược chọn mẫu, đặc biệt trong môi trường luồng hoặc phân tán, nơi bộ nhớ hoặc thông tin toàn cục bị giới hạn. Thuật toán chọn mẫu ngẫu nhiên hoặc chọn ngẫu nhiên một phần tử có biên lợi ích cao là ví dụ tiêu biểu cho hướng tiếp cận này. Trong bài toán tối đa hàm submodular không đơn điệu, thuật toán tham lam ngẫu nhiên chọn ngẫu nhiên phần tử từ một tập ứng viên có cận biên lợi ích không âm tại mỗi bước và đã chứng minh đạt hệ số xấp xỉ $\frac{1}{e}$ [32].

Mặc dù ngẫu nhiên hóa mang lại hiệu suất cao trong nhiều bài toán, trong một số trường hợp, có thể chuyển đổi thuật toán ngẫu nhiên thành thuật toán tất định tương đương thông qua quá trình gọi là khử ngẫu nhiên (derandomization). Kỹ thuật này sử dụng kỳ vọng có điều kiện (method of conditional expectation) hoặc ước lượng bi quan (pessimistic estimator) để xây dựng thuật toán tất định vẫn đảm bảo cùng hiệu suất như bản ngẫu nhiên. Tuy nhiên, quá trình khử ngẫu nhiên thường đòi hỏi phân tích sâu hơn và có thể làm tăng chi phí tính toán.

Nhìn chung, phương pháp ngẫu nhiên hóa là một công cụ linh hoạt và mạnh mẽ trong thiết kế thuật toán xấp xỉ, không chỉ vì khả năng đơn giản hóa việc ra quyết định mà còn vì hiệu quả lý thuyết mà nó mang lại. Trong luận án, kỹ thuật này được sử dụng như một thành phần quan trọng khi thiết kế các thuật toán xấp xỉ cho bài toán tối ưu hàm submodular không đơn điệu.

c) Phương pháp song song

Thuật toán song song (parallel algorithms) là một nhánh quan trọng trong thiết kế thuật toán hiện đại, đặc biệt trong bối cảnh xử lý dữ liệu lớn, hệ thống nhiều lõi (multi-core), hoặc tính toán hiệu năng cao (HPC). Khi các thuật toán tuần tự gặp giới hạn về tài nguyên hoặc thời gian xử lý, việc khai thác khả năng thực thi đồng thời trên nhiều đơn vị xử lý trở thành một hướng tiếp cận thiết yếu nhằm rút ngắn thời gian thực thi và nâng cao khả năng mở rộng.

Trong thiết kế thuật toán song song, một mô hình kinh điển được sử dụng rộng rãi là *PRAM* (Parallel Random Access Machine), bao gồm ba biến thể chính: EREW (Exclusive Read Exclusive Write), CREW (Concurrent Read Exclusive Write) và CRCW (Concurrent Read Concurrent Write). Đây là mô hình trừu tượng trong đó các bộ xử lý hoạt động đồng thời và có thể truy cập vào một vùng bộ nhớ chung. Ngoài ra, các mô hình

song song bộ nhớ chia sẻ (shared-memory) hoặc bộ nhớ phân tán (distributed-memory) cũng thường được sử dụng trong triển khai thực tế.

Một số kỹ thuật thiết kế thuật toán thường được khai thác hiệu quả trong phát triển thuật toán song song gồm:

- Chia để trị song song (parallel divide-and-conquer) [119]: Là chiến lược thiết kế thuật toán trong đó bài toán lớn được phân tách thành các bài toán con độc lập, được xử lý đồng thời, sau đó hợp nhất kết quả. Kỹ thuật này đặc biệt hữu ích trong môi trường bộ nhớ phân tán và thường được ứng dụng trong các bài toán sắp xếp, xử lý ma trận và các phép tính hình học.

- Pointer jumping [54]: Là kỹ thuật nền tảng trong lập trình song song, đặc biệt hiệu quả với danh sách liên kết và cây. Mỗi bước thực hiện đồng thời việc cập nhật các con trỏ, rút ngắn độ sâu của cấu trúc dữ liệu và từ đó giảm chiều sâu tính toán xuống $O(\log n)$ trong mô hình PRAM.

- Kỹ thuật Euler Tour [25]: Dựa trên việc duyệt cây theo chu trình Euler, kỹ thuật này chuyển đổi cấu trúc cây sang dạng tuyến tính, từ đó cho phép giải quyết các bài toán như tìm tổ tiên chung gần nhất (LCA), tính chiều sâu, hoặc đường đi trong cây bằng các thao tác mảng song song.

- Prefix computation (scan) [55]: Là kỹ thuật trung tâm trong thiết kế thuật toán song song, thực hiện phép tích lũy (như cộng, nhân, hoặc hợp logic) trên mảng đầu vào. Được coi là cấu trúc cơ bản trong nhiều hệ thống tính toán song song, nó là thành phần chủ lực trong lọc, phân phối chỉ số và tích lũy song song trên GPU cũng như CPU.

Để đánh giá hiệu quả thuật toán song song, sẽ sử dụng mô hình work-depth, trong đó hai đại lượng chính được quan tâm:

- Work (W): tổng số phép toán mà toàn bộ thuật toán thực hiện – tương ứng với thời gian chạy trong mô hình tuần tự.

- Depth (D): số vòng tính toán tuần tự bắt buộc – tương đương với thời gian chạy nếu có vô hạn bộ xử lý.

Trong bối cảnh này, công thức Brent cho phép ước lượng thời gian thực thi với P bộ xử lý như sau: $T_P = \mathcal{O}(\frac{W}{P} + D)$ trong đó T_P là thời gian thực thi thực tế. Công thức này minh hoạ rõ ràng sự đánh đổi giữa tổng công việc và mức độ song song hóa, đồng thời là công cụ chuẩn để tối ưu hiệu năng thực tế của thuật toán.

Về tiêu chí đánh giá, ngoài W và D, các yếu tố như tốc độ tăng tốc (speedup), hiệu năng mở rộng (scalability), cân bằng tải (load balancing) và chi phí truyền thông (communication overhead) cũng đóng vai trò quan trọng trong triển khai thực tế. Đặc biệt, một thuật toán song song được xem là hiệu quả nếu công việc tổng W không vượt quá nhiều so với phiên bản tuần tự tối ưu. Ngoài ra, luận án sử dụng thêm tiêu chí "độ phức tạp song song" để so sánh hiệu quả giữa các thuật toán song song, được định nghĩa như sau và sẽ được áp dụng trong Mục **2.4.**:

Định nghĩa 1.10 (Độ phức tạp song song [7])**.** Với một hàm mục tiêu f, độ phức tạp song song của một thuật toán là số vòng lặp tối thiểu cần thiết sao cho trong mỗi vòng lặp, thuật toán thực hiện $O(\text{poly}(n))$ truy vấn độc lập đến hàm mục tiêu, trong đó n là kích thước dữ liệu đầu vào của thuật toán.

Trong bối cảnh tối ưu tổ hợp, các nguyên lý trên cho phép thiết kế nhiều thuật toán hiệu quả cho các bài toán như: Liệt kê tập con hoặc tổ hợp; Tối đa submodular đơn điệu với ràng buộc lực lượng; Phân cụm (k-means, k-center), matching, hoặc phân hoạch cây. Các thuật toán tham lam cũng có thể được song song hóa một phần, đặc biệt khi phép chọn phần tử có thể thực hiện cục bộ. Một số nghiên cứu gần đây cho thấy có thể đạt được tỉ lệ xấp xỉ gần với tham lam tuần tự với độ phức tạp song song $\mathcal{O}(\log n)$ thông qua kỹ thuật chọn mẫu hoặc phân tách độc lập.

Tóm lại, phương pháp song song là một thành phần quan trọng trong thiết kế thuật toán xấp xỉ hiện đại. Khi được kết hợp với các kỹ thuật như ngẫu nhiên hóa hoặc luồng, nó mở ra khả năng thiết kế các thuật toán hiệu quả cả về lý thuyết và thực tiễn trong môi trường tính toán hiệu năng cao. Trong luận án, tư tưởng chia để trị song song được sử dụng để giảm thời gian xử lý trong khi vẫn duy trì bảo đảm xấp xỉ cho bài toán tối ưu hàm submodular không đơn điệu.

d) Phương pháp luồng

Theo mô hình dữ liệu luồng (data streaming) chuẩn [92], ta xem dữ liệu đầu vào như một tín hiệu một chiều $A : [1..n] \to \mathbb{R}$, được hình thành bởi một chuỗi các cập nhật $a_1, a_2, \ldots$ đến theo thứ tự thời gian. Mỗi cập nhật a_i tác động lên một toạ độ của A và tuỳ theo cách diễn giải cập nhật này, ta thu được ba biến thể mô hình điển hình: (i) mô hình chuỗi thời gian (time series), trong đó mỗi phần tử của luồng chính là giá trị $A[i]$ tại một chỉ số/thời điểm i; (ii) mô hình máy tính tiền (cash register), trong đó mỗi phần tử $a_i = (j, I_i)$ tương ứng với một gia tăng không âm I_i lên toạ độ j, tức $A[j] \leftarrow A[j] + I_i$; và (iii) mô hình turnstile, trong đó các cập nhật $a_i = (j, U_i)$ có thể mang giá trị dương hoặc âm và do đó cho phép cả thêm lẫn bớt, với biến thể strict turnstile yêu cầu $A[j] \geq 0$ tại mọi thời điểm. Ba mô hình này tạo thành một hệ phân cấp tự nhiên, trong đó turnstile là tổng quát nhất và bao hàm hai mô hình còn lại như các trường hợp đặc biệt. Dưới góc nhìn tài nguyên, mô hình dữ liệu luồng nhấn mạnh ba thước đo chính khi thiết kế thuật toán: thời gian xử lý mỗi phần tử trong luồng, dung lượng bộ nhớ (space) cần duy trì để mã hoá trạng thái hiện tại của tín hiệu A và thời gian tính toán giá trị của hàm quan tâm trên trạng thái này; mục tiêu lý tưởng là cả ba đại lượng này đều dưới tuyến tính theo kích thước miền dữ liệu n và số cập nhật đã quan sát t, lý tưởng là chỉ phụ thuộc đa thức theo $\log n$ và $\log t$.

Trên cơ sở mô hình dữ liệu đó, phương pháp luồng (streaming algorithms) được xem là một trong những hướng tiếp cận hiện đại và hiệu quả trong thiết kế thuật toán

xấp xỉ, đặc biệt trong bối cảnh xử lý dữ liệu lớn hoặc luồng dữ liệu động. Khác với các mô hình tính toán cổ điển, nơi thuật toán có toàn quyền truy cập và lưu trữ toàn bộ dữ liệu, trong mô hình luồng, thuật toán phải xử lý đầu vào đến theo từng phần tử, sử dụng bộ nhớ bị giới hạn nghiêm ngặt và thường chỉ được phép quét dữ liệu một lần hoặc rất ít lần. Chính những ràng buộc về tài nguyên này đã tạo ra nhu cầu phát triển các thuật toán mới với cấu trúc nhẹ, khả năng cập nhật theo thời gian thực và vẫn giữ được độ chính xác xấp xỉ nhất định. Một đặc điểm nổi bật của thuật toán luồng là không thể thực hiện các phép tính toàn cục phụ thuộc vào toàn bộ tập dữ liệu đã đọc. Điều này buộc thuật toán phải dựa vào các chiến lược chọn lọc và khai thác thông tin cục bộ, chẳng hạn như chọn mẫu đại diện, duy trì cấu trúc nén (sketch), hoặc xây dựng nghiệm từng bước theo nguyên tắc tham lam một lượt. Phương pháp này tỏ ra đặc biệt phù hợp với các bài toán mà mỗi phần tử trong luồng có thể được đánh giá độc lập hoặc có đóng góp biên có thể xấp xỉ theo thời gian. Chính thức, thuật toán luồng được định nghĩa như sau:

Định nghĩa 1.11 (Thuật toán luồng – Streaming Algorithm [1])**.** Thuật toán luồng là một thuật toán xấp xỉ xử lý dữ liệu được trình bày dưới dạng chuỗi, trong đó mỗi phần tử chỉ được truy cập một hoặc vài lần. Các đặc trưng chính gồm:

- **Số lượt quét:** Số lần thuật toán cần duyệt qua toàn bộ tập cơ sở.
- **Độ phức tạp bộ nhớ:** Lượng bộ nhớ cần dùng, thường tỉ lệ logarit theo kích thước dữ liệu.
- **Độ phức tạp truy vấn:** Số lượng truy vấn đến hàm mục tiêu f.

Một lớp bài toán tiêu biểu có thể khai thác hiệu quả thuật toán luồng là bài toán tối đa hàm submodular đơn điệu với ràng buộc lực lượng. Trong thiết lập này, thuật toán tham lam luồng hoạt động bằng cách duy trì một tập ứng viên có kích thước hữu hạn, cập nhật dần theo từng phần tử trong luồng và đảm bảo đạt hệ số xấp xỉ $\frac{1}{2} - \epsilon$ với chi phí bộ nhớ tuyến tính theo k và logarit theo kích thước dữ liệu, trong đó ϵ là tham số chính xác, k là ràng buộc lực lượng. Một ví dụ khác là bài toán chọn k phần tử có giá trị lớn nhất (top-k), trong đó thuật toán luồng duy trì một heap nhỏ chứa k phần tử tốt nhất hiện thời và cập nhật theo từng phần tử mới trong luồng.

Một số nguyên lý cơ bản trong thiết kế thuật toán luồng bao gồm các kỹ thuật như lấy mẫu ngẫu nhiên (reservoir sampling), xấp xỉ phân phối (hash-based counting) và sketching (CountSketch, AMS Sketch, Tug-of-War). Những kỹ thuật này không chỉ được áp dụng cho các bài toán tối ưu tổ hợp cổ điển mà còn cho các bài toán đếm, ước lượng số lượng phần tử phân biệt, tổng trọng số, hoặc moment bậc cao. Trong nhiều trường hợp, thuật toán chỉ cần bộ nhớ $\mathcal{O}(\log n)$ hoặc $\mathcal{O}(\text{poly}(\epsilon^{-1}))$ và đạt được sai số nhỏ với xác suất cao, thông qua việc sử dụng các bất đẳng thức xác suất như Markov hoặc Chebyshev. Hiệu suất của thuật toán luồng do đó được đánh giá không chỉ qua hệ số xấp xỉ mà còn dựa trên các tiêu chí đặc thù: (i) bộ nhớ sử dụng, (ii) số lượt quét qua dữ liệu

(1-pass hoặc multi-pass), (iii) thời gian cập nhật mỗi phần tử (update time) và (iv) xác suất sai số; đây là những yếu tố đặc biệt quan trọng trong thực tế khi triển khai thuật toán trên luồng dữ liệu tốc độ cao hoặc trong môi trường tính toán phân tán.

Mặc dù mang lại nhiều lợi ích, phương pháp luồng cũng tồn tại những giới hạn nhất định. Với các bài toán có tính toàn cục cao hoặc phụ thuộc vào cấu trúc kết hợp phức tạp (chẳng hạn như hàm submodular không đơn điệu với ràng buộc cấu trúc), việc xây dựng thuật toán luồng với đảm bảo lý thuyết tốt là rất khó khăn. Trong một số trường hợp, đã chứng minh được các giới hạn bất khả thi (impossibility bounds) về bộ nhớ và độ chính xác. Điều này đặt ra nhu cầu kết hợp với các kỹ thuật khác như ngẫu nhiên hóa, khử ngẫu nhiên, hoặc học trực tuyến để vượt qua những rào cản về tính toán trong môi trường tài nguyên hạn chế.

Tóm lại, thuật toán luồng là một công cụ quan trọng trong thiết kế thuật toán xấp xỉ hiện đại, đặc biệt trong bối cảnh xử lý dữ liệu quy mô lớn và luồng thông tin động. Với những đặc trưng riêng biệt và tiêu chí đánh giá khác biệt, phương pháp này bổ sung một hướng tiếp cận hữu hiệu và thiết thực cho bài toán tối ưu tổ hợp trong điều kiện tính toán hạn chế. Trong khuôn khổ luận án, phương pháp luồng được khai thác cho các bài toán mà dữ liệu đầu vào lớn hoặc không thuận tiện lưu trữ toàn bộ, từ đó làm nổi bật vai trò của số lượt duyệt dữ liệu, bộ nhớ sử dụng và số truy vấn oracle trong đánh giá thuật toán.

Các phương pháp thiết kế thuật toán xấp xỉ trên cho thấy mỗi hướng tiếp cận đều có ưu điểm và hạn chế riêng, nên không có một chiến lược duy nhất phù hợp cho mọi bài toán. Việc lựa chọn kỹ thuật thiết kế phụ thuộc vào cấu trúc hàm mục tiêu, loại ràng buộc và điều kiện thực thi, chẳng hạn bộ nhớ, thời gian xử lý hoặc môi trường song song. Cách nhìn này là cơ sở để luận án lựa chọn và kết hợp các kỹ thuật phù hợp cho từng bài toán nghiên cứu.

1.2. Giới thiệu về hàm submodular

1.2.1. Định nghĩa hàm submodular

Trong các bài toán tối ưu tổ hợp, việc khai thác cấu trúc đặc biệt của hàm mục tiêu đóng vai trò then chốt trong thiết kế các thuật toán hiệu quả. Một trong những lớp hàm quan trọng và phổ biến là hàm submodular, đại diện cho hiện tượng lợi ích biên giảm dần – một thuộc tính tự nhiên trong nhiều quá trình ra quyết định, như lan truyền ảnh hưởng, chọn cảm biến, hoặc tóm tắt dữ liệu.

Định nghĩa 1.12 (Định nghĩa hàm submodular [94])**.** Cho một tập hữu hạn V có n phần tử. Một hàm tập hợp $f : 2^V \to \mathbb{R}$ được gọi là *hàm submodular* nếu với mọi $A \subseteq B \subseteq V$ và $e \in V \setminus B$, ta có $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$.

Định nghĩa trên phản ánh trực tiếp nguyên lý lợi ích biên giảm dần của hàm submodular. Cụ thể, khi $A \subseteq B$, tập A có thể được xem là một tập lựa chọn nhỏ hơn, còn B là một tập lựa chọn đã chứa nhiều phần tử hơn. Khi đó, bất đẳng thức trong Định nghĩa 1.12 cho thấy lợi ích biên thu được khi thêm phần tử e vào tập nhỏ A không nhỏ hơn lợi ích biên khi thêm cùng phần tử đó vào tập lớn hơn B. Nói cách khác, giá trị đóng góp thêm của một phần tử có xu hướng giảm khi tập nền đã lớn hơn.

Ngoài cách diễn giải thông qua lợi ích biên giảm dần, tính submodular còn có một dạng tương đương thường dùng dựa trên hợp và giao của hai tập con. Cụ thể, một hàm tập $f : 2^V \to \mathbb{R}$ là submodular nếu và chỉ nếu với mọi $A, B \subseteq V$, ta có

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

Trong đó, $f(A) + f(B)$ biểu diễn tổng giá trị của hai tập lựa chọn khi xét riêng rẽ, còn $f(A \cup B) + f(A \cap B)$ kết hợp giá trị của phần hợp và phần giao của hai tập. Bất đẳng thức này thể hiện rằng khi hai tập có phần chồng lấp, việc gộp thông tin từ hai tập không tạo ra giá trị cộng thêm vượt quá tổng giá trị khi xét chúng riêng biệt. Đây là một cách nhìn khác của hiện tượng lợi ích biên giảm dần.

Trong nhiều ứng dụng thực tế, chẳng hạn như lựa chọn cảm biến, lan truyền ảnh hưởng hoặc phân bổ tài nguyên, hàm mục tiêu không chỉ thỏa mãn tính submodular mà còn có xu hướng tăng dần theo kích thước của tập đầu vào. Điều này phản ánh một thuộc tính tự nhiên trong các hệ thống tích lũy: việc thêm phần tử vào một tập hợp không làm giảm giá trị của hàm mục tiêu. Do đó, bên cạnh lớp hàm submodular tổng quát, các nghiên cứu về tối ưu submodular cũng đặc biệt quan tâm tới lớp hàm submodular đơn điệu không giảm.

Định nghĩa 1.13 (Hàm đơn điệu không giảm)**.** Cho V là một tập hữu hạn. Một hàm tập $f : 2^V \to \mathbb{R}$ được gọi là *đơn điệu không giảm* nếu với mọi $S \subseteq T \subseteq V$, ta có

$$f(S) \leq f(T).$$

Khi một hàm tập f vừa là hàm submodular, vừa là hàm đơn điệu không giảm, ta gọi f là *hàm submodular đơn điệu không giảm*. Trong trường hợp không gây nhầm lẫn, có thể gọi ngắn gọn là *hàm submodular đơn điệu*.

Trong toàn bộ luận án này, nếu không nói gì thêm, các hàm submodular được xét luôn được giả thiết là không âm và chuẩn hóa, nghĩa là $f(S) \geq 0,\ \forall S \subseteq V$, và $f(\emptyset) = 0$.

1.2.2. Biến thể hàm submodular

Để tăng khả năng mô hình hóa các bài toán thực tiễn, nhiều dạng mở rộng của hàm submodular đã được nghiên cứu và phát triển. Trong khuôn khổ của luận án này, nghiên cứu sinh tập trung vào việc giới thiệu hai biến thể mở rộng tiêu biểu và được ứng dụng rộng rãi của hàm submodular, đó là k-submodular và DR-submodular.

1.2.2.1. Định nghĩa hàm k-submodular

Trong các ứng dụng thực tiễn như phân loại đa nhãn, lựa chọn nhóm trong hệ thống đa tác nhân và tối đa ảnh hưởng theo nhiều chủ đề, thường yêu cầu là chia tập cơ sở thành nhiều nhóm rời nhau sao cho hàm mục tiêu đạt giá trị tối ưu. Trong bối cảnh đó, hàm submodular truyền thống, vốn chỉ làm việc trên các tập con, không đủ khả năng mô hình hóa các trường hợp cần chia tập thành k nhóm loại trừ lẫn nhau. Để khắc phục giới hạn này, khái niệm k-submodular đã được giới thiệu lần đầu tiên bởi Huber và Kolmogorov (2012) [61], sau đó Ward và cộng sự (2014) [123] là những người hoàn thiện lý thuyết về hàm k-submodular. Hàm k-submodular như một sự mở rộng tự nhiên của hàm submodular sang miền định nghĩa $(k+1)^V$, trong đó mỗi phần tử của tập cơ sở có thể được gán cho đúng một trong k nhóm hoặc không gán vào nhóm nào. Lớp hàm này giữ lại tính chất lợi ích biên giảm dần, đồng thời mở rộng tính ứng dụng sang nhiều bài toán tối ưu tổ hợp có cấu trúc phức tạp hơn. Hàm k-submodular được định nghĩa như sau:

Định nghĩa 1.14 (Định nghĩa hàm k-submodular [123])**.** Cho một tập cơ sở V và một số nguyên dương k, quy ước $[k] = \{1, 2, \ldots, k\}$ và $(k+1)^V = \{(S_1, S_2, \ldots, S_k) \mid S_i \subseteq V,\ \forall i \in [k],\ S_i \cap S_j = \emptyset,\ \forall i \neq j\}$ là họ k tập đôi một không giao nhau, được gọi là k-tập (k-set). Với $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$, ký hiệu $\mathrm{supp}(\mathbf{x}) = \bigcup_{i=1}^{k} X_i$. Với $e \notin \mathrm{supp}(\mathbf{x})$ và $i \in [k]$, lợi ích biên khi thêm cặp (e, i) vào $\mathbf{x}$ được định nghĩa bởi $\Delta_{(e,i)} f(\mathbf{x}) = f(X_1, \ldots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \ldots, X_k) - f(X_1, \ldots, X_k)$.

Hàm $f : (k+1)^V \to \mathbb{R}$ được gọi là *k-submodular* nếu thỏa mãn hai điều kiện sau:

1. Với mọi $\mathbf{x} = (X_1, \ldots, X_k), \mathbf{y} = (Y_1, \ldots, Y_k) \in (k+1)^V$ sao cho $X_i \subseteq Y_i$ với mọi $i \in [k]$, mọi $e \notin \mathrm{supp}(\mathbf{y})$ và mọi $i \in [k]$, ta có $\Delta_{(e,i)} f(\mathbf{x}) \geq \Delta_{(e,i)} f(\mathbf{y})$.
2. Với mọi $\mathbf{x} \in (k+1)^V$, mọi $e \notin \mathrm{supp}(\mathbf{x})$ và mọi $i, j \in [k]$, $i \neq j$, ta có $\Delta_{(e,i)} f(\mathbf{x}) + \Delta_{(e,j)} f(\mathbf{x}) \geq 0$.

Điều kiện thứ nhất trong Định nghĩa 1.14 là dạng mở rộng của nguyên lý lợi ích biên giảm dần trong hàm submodular cổ điển. Cụ thể, nếu $\mathbf{x}$ là một nghiệm nhỏ hơn $\mathbf{y}$ theo nghĩa $X_i \subseteq Y_i$ với mọi $i \in [k]$, thì lợi ích biên khi thêm cùng một cặp (e, i) vào $\mathbf{x}$ không nhỏ hơn lợi ích biên khi thêm cặp đó vào $\mathbf{y}$. Điều kiện này thể hiện rằng khi nghiệm hiện tại đã chứa nhiều phần tử hơn ở từng thành phần, đóng góp bổ sung của một phần tử mới không tăng lên.

Điều kiện thứ hai phản ánh đặc trưng riêng của miền $(k+1)^V$. Trong miền này, mỗi phần tử $e \in V$ có thể không được chọn, hoặc được gán vào đúng một trong k thành phần. Do đó, với một phần tử chưa thuộc $\mathrm{supp}(\mathbf{x})$, các lựa chọn thêm (e, i) và (e, j) với $i \neq j$ là loại trừ lẫn nhau. Bất đẳng thức $\Delta_{(e,i)} f(\mathbf{x}) + \Delta_{(e,j)} f(\mathbf{x}) \geq 0$ kiểm soát quan hệ giữa hai lợi ích biên tương ứng với hai thành phần khác nhau của cùng một phần tử. Đây là điểm khác biệt quan trọng giữa hàm k-submodular và hàm submodular cổ điển.

Trong trường hợp đặc biệt $k = 1$, miền $(k+1)^V$ tương ứng với miền các tập con của V. Khi đó, điều kiện thứ hai không còn xuất hiện, còn điều kiện thứ nhất trở thành nguyên lý lợi ích biên giảm dần của hàm submodular cổ điển. Vì vậy, hàm k-submodular có thể được xem là một mở rộng tự nhiên của hàm submodular từ miền tập con sang miền các k-tập.

Hàm k-submodular cho phép mô hình hóa các bài toán trong đó mỗi phần tử có thể không được chọn, hoặc được gán vào một trong k nhóm khác nhau. Ví dụ:

- Tối đa lan truyền ảnh hưởng đa chủ đề: mỗi nút có thể được chọn làm hạt giống cho một trong k chủ đề, và các chủ đề có thể có cơ chế lan truyền riêng.
- Phân nhãn dữ liệu đa lớp: mỗi phần tử trong tập dữ liệu có thể được gán vào một trong k nhãn cho trước, trong đó các nhãn là loại trừ lẫn nhau.

Định nghĩa 1.15 (Hàm k-submodular đơn điệu không giảm)**.** Cho $f : (k+1)^V \to \mathbb{R}$ là một hàm k-submodular. Hàm f được gọi là *đơn điệu không giảm* nếu với mọi $\mathbf{x} \in (k+1)^V$, $e \notin \text{supp}(\mathbf{x})$ và $i \in [k]$, ta có $\Delta_{(e,i)} f(\mathbf{x}) \geq 0$.

Điều kiện trong Định nghĩa 1.15 có nghĩa là việc thêm một phần tử chưa được chọn vào bất kỳ thành phần hợp lệ nào không làm giảm giá trị của hàm mục tiêu. Cần lưu ý rằng tính k-submodular và tính đơn điệu không giảm là hai thuộc tính khác nhau; một hàm k-submodular không nhất thiết phải đơn điệu không giảm.

Tương tự như đối với hàm submodular, trong toàn bộ luận án này, nếu không nói gì thêm, các hàm k-submodular được xét luôn được giả thiết là không âm và chuẩn hóa, nghĩa là $f(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in (k+1)^V$, và $f(\emptyset, \ldots, \emptyset) = 0$.

1.2.2.2. Định nghĩa hàm DR-submodular

Hàm submodular cổ điển có thể được định nghĩa trên không gian tập con nhị phân $\{0,1\}^V$, với V là tập cơ sở. Trong mô hình này, mỗi phần tử $e \in V$ chỉ có hai trạng thái: được chọn hoặc không được chọn vào lời giải. Các bài toán tối ưu hàm submodular cổ điển đã chứng tỏ hiệu quả trong nhiều bài toán tối ưu tổ hợp, chẳng hạn như tóm tắt dữ liệu, đặt cảm biến, tối đa hóa doanh thu,.... Tuy nhiên, trong nhiều bối cảnh thực tế, mỗi phần tử $e \in V$ có thể được lựa chọn nhiều lần, ví dụ như phân bổ nhiều đơn vị tài nguyên cho cùng một tác vụ. Khi đó, mô hình nhị phân thuần túy trở nên hạn chế vì không phản ánh đầy đủ sự thay đổi của lợi ích biên theo số lượng đơn vị được lựa chọn. Để khắc phục hạn chế này, Soma và cộng sự [112] đã nghiên cứu các mở rộng của tính submodular trên không gian lưới số nguyên không âm $\mathbb{Z}_+^V$, trong đó mỗi tọa độ biểu diễn số lần hoặc số đơn vị được chọn của một phần tử.

Cho V là một tập cơ sở hữu hạn. Xét hàm $f : \mathbb{Z}_+^V \to \mathbb{R}$, ánh xạ mỗi vectơ nguyên không âm trên V tới một số thực. Với mỗi $e \in V$, vectơ đơn vị $\mathbf{1}_e \in \mathbb{Z}_+^V$ được xác định bởi $\mathbf{1}_e(t) = 1$ nếu $t = e$ và $\mathbf{1}_e(t) = 0$ nếu $t \neq e$. Với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, lợi ích biên của f khi

cộng thêm $\mathbf{y}$ vào $\mathbf{x}$ được ký hiệu bởi $f(\mathbf{y} \mid \mathbf{x}) = f(\mathbf{x}+\mathbf{y}) - f(\mathbf{x})$. Ngoài ra, ký hiệu $\{\mathbf{x}\}$ là tập các phần tử $e \in V$ sao cho $\mathbf{x}(e) > 0$.

Định nghĩa 1.16 (Định nghĩa hàm DR-submodular [112])**.** Một hàm $f : \mathbb{Z}_+^V \to \mathbb{R}$ được gọi là *diminishing-return submodular*, hay *DR-submodular*, nếu với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$ sao cho $\mathbf{x} \leq \mathbf{y}$, và với mọi $e \in V$, ta có $f(\mathbf{x} + \mathbf{1}_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \mathbf{1}_e) - f(\mathbf{y})$.

Định nghĩa 1.16 mở rộng trực tiếp nguyên lý lợi ích biên giảm dần từ hàm submodular trên miền tập con sang hàm xác định trên lưới nguyên không âm. Cụ thể, nếu $\mathbf{x} \leq \mathbf{y}$, tức là $\mathbf{x}(e) \leq \mathbf{y}(e)$ với mọi $e \in V$, thì $\mathbf{y}$ biểu diễn một trạng thái đã chứa không ít đơn vị hơn $\mathbf{x}$ ở mọi tọa độ. Khi đó, bất đẳng thức trong Định nghĩa 1.16 cho thấy lợi ích biên khi tăng thêm một đơn vị tại tọa độ e ở trạng thái nhỏ hơn $\mathbf{x}$ không nhỏ hơn lợi ích biên khi tăng cùng một đơn vị đó ở trạng thái lớn hơn $\mathbf{y}$.

Từ định nghĩa trên, bằng quy nạp theo k, ta suy ra với mọi $k \in \mathbb{Z}_+$, $\mathbf{x} \leq \mathbf{y}$ và $e \in V$, $f\big(k\mathbf{1}_e \mid \mathbf{x}\big) \geq f\big(k\mathbf{1}_e \mid \mathbf{y}\big)$. Điều này có nghĩa là khi cộng thêm nhiều đơn vị của cùng một phần tử, tổng lợi ích biên thu được tại một trạng thái nhỏ hơn vẫn không nhỏ hơn tổng lợi ích biên tại một trạng thái lớn hơn.

Định nghĩa 1.17 (Tính chất submodular trên lưới nguyên [112])**.** Một hàm $f : \mathbb{Z}_+^V \to \mathbb{R}$ được gọi là *submodular trên lưới nguyên* nếu với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, ta có

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}),$$

trong đó: $(\mathbf{x} \vee \mathbf{y})(e) = \max\{\mathbf{x}(e), \mathbf{y}(e)\}$, $(\mathbf{x} \wedge \mathbf{y})(e) = \min\{\mathbf{x}(e), \mathbf{y}(e)\}$.

Tính submodular trên lưới nguyên trong Định nghĩa 1.17 là dạng tương tự của bất đẳng thức submodular cổ điển khi thay phép hợp và giao của hai tập bởi phép lấy cực đại và cực tiểu theo từng tọa độ. Cụ thể, $\mathbf{x} \vee \mathbf{y}$ đóng vai trò tương tự như phép hợp, còn $\mathbf{x} \wedge \mathbf{y}$ đóng vai trò tương tự như phép giao. Do đó, bất đẳng thức

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y})$$

có thể được xem là một mở rộng của bất đẳng thức $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ trong trường hợp hàm tập.

Cần lưu ý rằng tính DR-submodular là một điều kiện mạnh hơn tính submodular trên lưới nguyên. Nói cách khác, nếu f là DR-submodular thì f cũng là submodular trên lưới nguyên, nhưng chiều ngược lại không nhất thiết đúng.

Định nghĩa 1.18 (Hàm DR-submodular đơn điệu không giảm)**.** Cho $f : \mathbb{Z}_+^V \to \mathbb{R}$ là một hàm DR-submodular. Hàm f được gọi là *đơn điệu không giảm* nếu với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$ sao cho $\mathbf{x} \leq \mathbf{y}$, ta có $f(\mathbf{x}) \leq f(\mathbf{y})$.

Điều kiện trong Định nghĩa 1.18 có nghĩa là việc tăng số đơn vị được chọn ở một hoặc nhiều tọa độ không làm giảm giá trị của hàm mục tiêu. Cần lưu ý rằng tính

DR-submodular và tính đơn điệu không giảm là hai thuộc tính khác nhau; một hàm DR-submodular không nhất thiết phải đơn điệu không giảm. Trong các phân tích thuật toán, nếu không gây nhầm lẫn, ta thường giả sử hàm f được chuẩn hóa, tức là $f(\mathbf{0}) = 0$. Khi f đồng thời đơn điệu không giảm và chuẩn hóa, ta có $f(\mathbf{x}) \geq 0$ với mọi $\mathbf{x} \in \mathbb{Z}_+^V$.

Tương tự như đối với hàm submodular cổ điển, trong toàn bộ luận án này, nếu không nói gì thêm, các hàm DR-submodular được xét luôn được giả thiết là không âm và chuẩn hóa, nghĩa là $f(\mathbf{x}) \geq 0,\ \forall \mathbf{x} \in \mathbb{Z}_+^V$, và $f(\mathbf{0}) = 0$.

Luận án trình bày và chứng minh hai bổ đề cơ bản về hàm DR-submodular, đóng vai trò nền tảng cho việc phân tích lý thuyết các thuật toán trên lưới nguyên. Hai bổ đề được phát biểu như sau:

Bổ đề 1.1. *Cho $f : \mathbb{Z}_+^V \to \mathbb{R}_+$ là một hàm DR-submodular. Với mọi $\boldsymbol{s} \in \mathbb{Z}_+^V$ và hai vector $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^V$ sao cho $\boldsymbol{x} \wedge \boldsymbol{y} = \mathbf{0}$, ta có: $f(\boldsymbol{s}) \leq f(\boldsymbol{s} \vee \boldsymbol{x}) + f(\boldsymbol{s} \vee \boldsymbol{y})$.*

Chứng minh. Vì f là một hàm DR-submodular và không âm, ta có:

$$\begin{aligned} f(\mathbf{s} \vee \mathbf{x}) + f(\mathbf{s} \vee \mathbf{y}) &\geq f(\mathbf{s} \vee \mathbf{x} \vee \mathbf{y}) + f((\mathbf{s} \vee \mathbf{x}) \wedge (\mathbf{s} \vee \mathbf{y})) && \text{(tính chất submodular)} \\ &= f(\mathbf{s} \vee \mathbf{x} \vee \mathbf{y}) + f(\mathbf{s}) && \text{(do điều kiện } \mathbf{x} \wedge \mathbf{y} = \mathbf{0}) \\ &\geq f(\mathbf{s}) && \text{(do } f \text{ không âm).} \end{aligned}$$

Điều này hoàn tất chứng minh. □

Bổ đề 1.2. *Cho $f : \mathbb{Z}_+^V \to \mathbb{R}_+$ là một hàm DR-submodular. Với mọi $\boldsymbol{x} \in \mathbb{Z}_+^V$ và một số nguyên $t \geq 0$, ta có: $f(t\mathbf{1}_e \mid \boldsymbol{x}) \leq t f(\mathbf{1}_e \mid \boldsymbol{x})$.*

Chứng minh. Do f là một hàm DR-submodular, ta có:

$$\begin{aligned} f(t\mathbf{1}_e \mid \mathbf{x}) &= f(\mathbf{x} + t\mathbf{1}_e) - f(\mathbf{x}) \\ &= f(\mathbf{1}_e \mid \mathbf{x} + (t-1)\mathbf{1}_e) + f(\mathbf{1}_e \mid \mathbf{x} + (t-2)\mathbf{1}_e) + \ldots + f(\mathbf{1}_e \mid \mathbf{x}) \\ &\leq t f(\mathbf{1}_e \mid \mathbf{x}). \end{aligned}$$

Điều này hoàn tất chứng minh. □

1.3. Các bài toán tiêu biểu tối ưu hàm submodular

Tùy theo mục tiêu tối ưu và dạng hàm mục tiêu, các bài toán tối ưu hàm submodular có thể được chia thành nhiều nhóm, trong đó hai nhóm gắn trực tiếp với luận án là bài toán tối đa hàm submodular hoặc các biến thể của nó, và bài toán phủ submodular hoặc các biến thể tương ứng. Mỗi nhóm bài toán đặc trưng cho một dạng mục tiêu khác nhau: bài toán tối đa nhằm chọn nghiệm có giá trị hàm lớn nhất dưới các ràng buộc cho trước, trong khi bài toán phủ hướng đến việc đạt một ngưỡng giá trị hàm với chi phí nhỏ nhất. Dưới đây là định nghĩa cụ thể của hai nhóm bài toán này.

Trong các bài toán tối ưu hàm submodular, ta thường giả định có truy cập oracle đến hàm mục tiêu. Điều này có nghĩa là, thay vì yêu cầu biết biểu thức tường minh của hàm submodular (hoặc k-submodular, DR-submodular), thuật toán chỉ được phép tương tác với f thông qua một "hộp đen": với mỗi tập con $S \subseteq V$ đưa vào, oracle trả về giá trị $f(S)$. Mỗi lần gọi như vậy được xem là một phép truy vấn (query), và số lượng truy vấn tới oracle thường được dùng làm một trong những thước đo chính cho độ phức tạp của thuật toán, bên cạnh chi phí tính toán khác. Nhờ làm việc trong mô hình oracle, các thuật toán được thiết kế có tính khái quát cao và có thể áp dụng trực tiếp cho nhiều bài toán khác nhau, miễn là hàm mục tiêu thỏa mãn tính submodular (hoặc các biến thể liên quan), không phụ thuộc vào dạng biểu thức tường minh cụ thể. Các bài toán trong luận án đều sử dụng giả định truy cập oracle đến hàm mục tiêu.

1.3.1. Bài toán tối đa hàm submodular

Bài toán tối đa hàm submodular là một trong những bài toán cơ bản và kinh điển trong tối ưu tổ hợp. Với mục tiêu tìm một tập con tối ưu từ một tập cơ sở hữu hạn sao cho giá trị của một hàm submodular được cực đại, bài toán này không chỉ mang ý nghĩa lý thuyết sâu sắc mà còn xuất hiện phổ biến trong nhiều ứng dụng thực tiễn như chọn cảm biến, tóm tắt dữ liệu và tối đa ảnh hưởng trên mạng xã hội, được phát biểu chính thức như sau:

Định nghĩa 1.19 (Tối đa hàm submodular với ràng buộc $\mathcal{I}$ [88])**.** Cho một tập cơ sở hữu hạn $V = \{e_1, \ldots, e_n\}$, một hàm submodular không âm $f : 2^V \to \mathbb{R}_+$ và một họ tập khả thi $\mathcal{I} \subseteq 2^V$ theo ràng buộc, khi đó bài toán được phát biểu như sau:

$$\max_{S \in \mathcal{I}} f(S).$$

Dựa trên phát biểu chính thức của bài toán tối đa hàm submodular với ràng buộc $\mathcal{I}$, nghiên cứu sinh phát biểu bài toán tối đa hàm k-submodular và DR-submodular như sau:

Định nghĩa 1.20 (Tối đa hàm k-submodular với ràng buộc $\mathcal{I}$)**.** Cho một tập cơ sở hữu hạn $V = \{e_1, \ldots, e_n\}$, một hàm $f : (k+1)^V \to \mathbb{R}_+$ là k-submodular và một tập các cấu hình khả thi $\mathcal{I} \subseteq (k+1)^V$, bài toán được phát biểu như sau:

$$\max_{\mathbf{s} \in \mathcal{I}} f(\mathbf{s}),$$

trong đó $\mathbf{s} = (S_1, \ldots, S_k)$ là các tập con rời nhau $S_i \subseteq V$, $S_i \cap S_j = \emptyset$ với mọi $i \neq j$.

Định nghĩa 1.21 (Tối đa hàm DR-submodular với ràng buộc $\mathcal{I}$)**.** Cho một tập cơ sở hữu hạn $V = \{e_1, \ldots, e_n\}$, một hàm DR-submodular không âm $f : \mathbb{Z}_+^V \to \mathbb{R}_+$ và một tập khả thi tổng quát $\mathcal{I} \subseteq \mathbb{Z}_+^V$, bài toán được phát biểu như sau:

$$\max_{\mathbf{x} \in \mathcal{I}} f(\mathbf{x}).$$

Bài toán tối đa hàm submodular – và các biến thể như k-submodular hay DR-submodular – đã được chứng minh là NP-khó [94]. Trước thực tế này, người ta chuyển sang thiết kế các thuật toán xấp xỉ với đảm bảo tỉ lệ hằng số, nhằm thu được nghiệm gần tối ưu trong thời gian đa thức. Đối với hàm submodular không nhất thiết đơn điệu, các phương pháp như tham lam kép hay các biến thể tìm kiếm lân cận cho phép đạt tỉ lệ xấp xỉ $1/2$ [15]. Trong khi đó, nếu hàm mục tiêu là submodular đơn điệu, tính chất lợi ích biên giảm dần được khai thác hiệu quả thông qua thuật toán tham lam tiêu chuẩn – có thể đảm bảo tỉ lệ $1 - 1/e \approx 0.632$ so với giá trị tối ưu – hoặc thông qua khung tham lam liên tục kết hợp làm tròn ngẫu nhiên cho bài toán với ràng buộc matroid [17]. Dù tỉ lệ xấp xỉ tăng đáng kể trong trường hợp đơn điệu, bài toán vẫn giữ nguyên tính NP-khó và do đó không có phương pháp đa thức cho nghiệm chính xác.

1.3.2. Bài toán phủ submodular

Bên cạnh bài toán tối đa giá trị hàm mục tiêu, một lớp bài toán quan trọng khác là bài toán phủ submodular (submodular Cover) – là bài toán đối ngẫu của bài toán tối đa hàm submodular. Thay vì tìm tập con có giá trị hàm lớn nhất, mục tiêu của bài toán phủ là tìm tập con có chi phí nhỏ nhất sao cho giá trị hàm đạt được một ngưỡng tối thiểu cho trước. Bài toán này thường xuất hiện trong các ứng dụng như giám sát cảm biến, bao phủ tập thông tin, hoặc bảo đảm chất lượng dịch vụ. Tùy vào dạng hàm submodular được sử dụng (chuẩn, DR-submodular, hoặc k-submodular), bài toán có thể biểu diễn trên nhiều không gian khác nhau nhưng cùng mục tiêu "đảm bảo ngưỡng với chi phí nhỏ nhất". Dưới đây là phát biểu chính thức của bài toán phủ submodular:

Định nghĩa 1.22 (Bài toán phủ submodular [8])**.** Cho một tập cơ sở $V = \{e_1, \ldots, e_n\}$, một hàm submodular $f : 2^V \to \mathbb{R}_+$ với $f(\emptyset) = 0$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí của một tập con $S \subseteq V$ là $c(S) = \sum_{e \in S} c(e)$. Với một ngưỡng chất lượng $T > 0$, mục tiêu là tìm tập $S \subseteq V$ sao cho: $f(S) \geq T \quad$ và $\quad c(S)$ là nhỏ nhất. Bài toán này được phát biểu như sau:

$$\min_{S \subseteq V, f(S) \geq T} c(S).$$

Dựa theo phát biểu chính thức của bài toán phủ submodular, luận án phát biểu bài toán phủ k-submodular và phủ DR-submodular như sau:

Định nghĩa 1.23 (Bài toán phủ DR-submodular)**.** Cho một tập cơ sở $V = \{e_1, \ldots, e_n\}$, một hàm DR-submodular $f : \mathbb{Z}_+^V \to \mathbb{R}_+$ với $f(\mathbf{0}) = 0$, một lưới nguyên bị chặn bởi $\mathbb{B}$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí của một vector $\mathbf{x} \in \mathbb{Z}_+^V$ là $c(\mathbf{x}) = \sum_{e \in V} c(e)\mathbf{x}(e)$. Với một ngưỡng $T > 0$, bài toán được phát biểu như sau:

$$\min_{\mathbf{x} \in \mathbb{Z}_+^V,\ f(\mathbf{x}) \geq T} c(\mathbf{x}).$$

Định nghĩa 1.24 (Bài toán phủ k-submodular)**.** Cho một tập cơ sở $V = \{e_1, \ldots, e_n\}$, một hàm k-submodular $f : (k+1)^V \to \mathbb{R}_+$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí của một k-tập $\mathbf{s} = (S_1, \ldots, S_k)$ là $c(\mathbf{s}) = \sum_{i=1}^{k} \sum_{e \in S_i} c(e)$. Với một ngưỡng $T > 0$, bài toán được phát biểu như sau:

$$\min_{\mathbf{s}=(S_1,\ldots,S_k),\ f(\mathbf{s}) \geq T} c(\mathbf{s}),$$

trong đó các tập $S_1, \ldots, S_k \subseteq V$ đôi một rời nhau.

Khi tất cả các phần tử $e \in V$ đều được gán chi phí dương $c(e) = 1$, lúc đó bài toán phủ submodular với chi phí nhỏ nhất suy biến thành bài toán phủ submodular với kích thước nhỏ nhất. Bài toán phủ submodular (hoặc các biến thể khác của hàm submodular) được chứng minh là NP-khó ngay cả trong trường hợp hàm f đơn điệu và chi phí là số nguyên dương [125]. Trong trường hợp hàm submodular không đơn điệu việc chứng minh bài toán có nghiệm hay không đã là một vấn đề phức tạp. Do đó, các nghiên cứu đều tập trung vào thiết kế các thuật toán xấp xỉ cho trường hợp hàm submodular đơn điệu. Phương pháp tham lam cổ điển, với tiêu chí chọn phần tử có tỉ lệ lợi ích biên trên chi phí lớn nhất, được chứng minh cho phép đạt tỉ lệ xấp xỉ $\ln(\Delta) + 1$, trong đó $\Delta = \max_{S \subseteq U} \max_{x \in U \setminus S} f(S \cup \{x\}) - f(S)$ [125]. Đáng chú ý, tỷ lệ xấp xỉ này là tốt nhất có thể đạt được trừ khi $P = NP$ [107]. Nhiều mở rộng của bài toán phủ submodular đã được đề xuất, chẳng hạn Bar-Ilan và các cộng sự [8] mở rộng bài toán này sang các bài toán tổng quát hơn như cây khung tối thiểu có đường kính bị chặn và các bài toán chọn trung tâm, đồng thời chứng minh rằng chúng vẫn NP-khó và chỉ có thể xấp xỉ trong tỉ lệ logarit (trừ khi $P = NP$). Những kết quả này khẳng định vị trí trung tâm của bài toán phủ submodular trong lớp các bài toán tối ưu tổ hợp khó và nhấn mạnh vai trò của các phương pháp xấp xỉ hiệu quả trong thực tiễn.

1.4. Các ràng buộc phổ biến của bài toán tối ưu hàm submodular

Trong các bài toán tối ưu submodular, ràng buộc giữ vai trò quan trọng trong việc mô hình hóa các giới hạn thực tế như ngân sách, số lượng tài nguyên hoặc cấu trúc lựa chọn. Sự kết hợp của các loại ràng buộc khác nhau có thể ảnh hưởng đáng kể đến cả độ phức tạp lẫn phương pháp giải. Trong phần này, luận án giới thiệu những dạng ràng buộc thường gặp trong tối ưu hàm submodular. Để minh họa, khuôn mẫu hàm submodular chuẩn được sử dụng nhờ tính khái quát và sự đơn giản trong biểu diễn công thức; các ràng buộc này cũng có thể được mở rộng cho các biến thể khác của hàm submodular.

1.4.1. Không ràng buộc

Không ràng buộc (unconstrained) [43] thường chỉ được nghiên cứu trong ngữ cảnh hàm submodular không đơn điệu. Trong trường hợp hàm đơn điệu, lời giải tối ưu sẽ luôn là toàn bộ tập V, do đó không mang nhiều ý nghĩa thực tiễn. Với hàm f không đơn

điệu, bài toán vẫn mang tính tối ưu và có độ phức tạp cao. Bài toán có dạng:

$$\max_{S \subseteq V} f(S),$$

trong đó f là hàm submodular không đơn điệu, V là tập cơ sở hữu hạn. Dạng bài toán này thường xuất hiện như một bài toán con trong quá trình thiết kế thuật toán cho các bài toán tối đa hàm submodular có ràng buộc.

1.4.2. Ràng buộc về lực lượng

Ràng buộc lực lượng (cardinality constraint hoặc size constraint) [94] giới hạn số lượng phần tử được chọn trong tập lời giải S. Đây là một ràng buộc phổ biến trong các bài toán tối ưu với giới hạn về số lượng tài nguyên. Bài toán có dạng:

$$\max_{S \subseteq V,\ |S| \leq k} f(S),$$

với $k \in \mathbb{Z}_+$ là số phần tử tối đa được phép chọn, V là tập cơ sở. Trong luận án, ràng buộc lực lượng là một dạng ràng buộc quan trọng khi mở rộng bài toán tối đa từ hàm tập hợp sang hàm *DR-submodular* trên lưới nguyên.

1.4.3. Ràng buộc về chi phí

Ràng buộc chi phí (knapsack constraint) [116] yêu cầu tổng chi phí của các phần tử được chọn không vượt quá một ngân sách cho trước B. Đây là mô hình phù hợp cho các ứng dụng thực tế như ngân sách tiếp thị, thời gian, hoặc tài nguyên tính toán. Bài toán có dạng:

$$\max_{S \subseteq V,\ c(S) \leq B} f(S),$$

trong đó $c : V \to \mathbb{R}_+$ là hàm chi phí, chi phí của một tập con $S \subseteq V$ là $c(S) = \sum_{e \in S} c(e)$ và $B > 0$ là ngân sách, V là tập cơ sở, mỗi một phần tử $e \in V$ được gán một chi phí dương $c(e) > 0$. Khi chi phí của tất cả các phần tử $e \in V$ đều được cài đặt bằng 1, lúc đó ràng buộc chi phí sẽ suy biến thành ràng buộc lực lượng.

Luận án nghiên cứu ràng buộc chi phí trong cả hai hướng: tối đa hàm mục tiêu dưới giới hạn ngân sách và phủ hàm mục tiêu với tổng chi phí nhỏ nhất. Ở hướng thứ nhất, chi phí đóng vai trò giới hạn tài nguyên được phép sử dụng; ở hướng thứ hai, chi phí chính là đại lượng cần tối thiểu hóa khi nghiệm phải đạt một ngưỡng giá trị cho trước.

Ngoài ra, một biến thể mở rộng của ràng buộc chi phí là ràng buộc chi phí nhóm, trong đó chi phí được xét theo từng nhóm hoặc từng loại lựa chọn. Dạng ràng buộc này phù hợp với các bài toán tối ưu hàm k-submodular, nơi mỗi phần tử có thể được gán vào một trong nhiều nhóm khác nhau.

1.4.4. Ràng buộc khác

Ngoài các ràng buộc cơ bản, bài toán tối ưu submodular còn có thể được mở rộng với các ràng buộc phức tạp hơn như:

- Ràng buộc cấu trúc (matroid constraint) [17]: mô tả bởi một họ độc lập $\mathcal{I} \subseteq 2^V$ thỏa mãn tính chất bắc cầu và tính mở rộng, trong đó V là tập cơ sở. Cụ thể, $\mathcal{I}$ thỏa mãn hai tính chất: (1) với mọi $A \subseteq B \subseteq V$ và $B \in \mathcal{I}$ thì suy ra $A \in \mathcal{I}$; (2) $A, B \in \mathcal{I}$ và nếu $|B| > |A|$ thì $\exists e \in B \setminus A$ sao cho $A \cup \{e\} \in \mathcal{I}$.

- Ràng buộc nhiều chi phí (multi-knapsack) [75]: mỗi phần tử e mang đồng thời m loại chi phí, được biểu diễn bởi các hàm $c_i(e)$ với $i \in [m]$. Ràng buộc yêu cầu tổng chi phí theo từng loại trong tập chọn S không vượt quá ngưỡng B_i đã cho ứng với mỗi $i \in [m]$:

$$\sum_{e \in S} c_i(e) \leq B_i, \quad \forall i \in [m].$$

Những ràng buộc này thường yêu cầu thiết kế các thuật toán chuyên biệt và đánh đổi giữa độ chính xác và hiệu năng.

1.5. Các ứng dụng tiêu biểu của bài toán tối ưu hàm submodular

Trong bối cảnh dữ liệu lớn, các thuật toán tối ưu hàm submodular tiếp tục phát triển theo hướng giảm độ phức tạp truy vấn đến hàm mục tiêu, đồng thời đạt được hệ số xấp xỉ tốt hơn. Những nghiên cứu gần đây cho thấy các thuật toán này có thể giải quyết hiệu quả nhiều bài toán thực tiễn, chẳng hạn như lan truyền ảnh hưởng trong mạng xã hội, tối đa doanh thu quảng cáo, tóm tắt dữ liệu quy mô lớn, hoặc phân phối tài nguyên trong hệ thống cảm biến. Đặc biệt, các biến thể nâng cao như k-submodular và DR-submodular đã mở rộng đáng kể khả năng mô hình hóa, giúp xử lý tốt hơn những trường hợp có ràng buộc phức tạp. Phần này trình bày một số bài toán tiêu biểu có liên quan trực tiếp đến các thực nghiệm trong luận án, bao gồm: tối đa ảnh hưởng trên mạng xã hội, tối đa doanh thu, Max-Cut, tóm tắt hình ảnh và đặt cảm biến.

1.5.1. Bài toán tối đa ảnh hưởng

Bài toán tối đa ảnh hưởng (influence maximization) là một trong những vấn đề trọng yếu trong phân tích mạng xã hội và khai phá dữ liệu. Bài toán này thường được sử dụng để mô hình hóa các chiến dịch marketing lan truyền, hệ thống gợi ý sản phẩm, lan truyền tin tức, hoặc kiểm soát dịch bệnh. Mục tiêu của bài toán là xác định một tập người dùng khởi tạo ban đầu trong mạng xã hội sao cho mức độ ảnh hưởng lan rộng từ họ là lớn nhất có thể, theo một mô hình lan truyền xác định. Hai mô hình lan truyền phổ biến nhất được sử dụng là mô hình lan truyền độc lập (Independent Cascade - IC) và mô hình ngưỡng tuyến tính (Linear Threshold - LT) [66]. Trong khuôn khổ của luận án này, luận án tập trung vào mô hình IC do tính mô phỏng xác suất trực quan và tính chất submodular rõ ràng. Bài toán được phát biểu như sau:

Định nghĩa 1.25 (Bài toán tối đa ảnh hưởng [66])**.** Cho một đồ thị có hướng $G = (V, E)$, đại diện cho mạng xã hội, trong đó mỗi cạnh $(u, v) \in E$ mang một xác suất lan truyền $p_{uv} \in [0, 1]$. Với mỗi tập khởi tạo $S \subseteq V$, hàm lan truyền $f(S)$ được định nghĩa là kỳ vọng số lượng nút bị ảnh hưởng khi bắt đầu quá trình lan truyền từ S theo mô hình IC. Bài toán tối đa ảnh hưởng được phát biểu như sau:

$$\max_{S \subseteq V, |S| \leq k} f(S)$$

trong đó k là ngân sách (số lượng người dùng có thể chọn để khởi tạo).

Bài toán này đã được chứng minh là NP-khó và thậm chí khó xấp xỉ trong một số trường hợp [66]. Kempe và các cộng sự [66] cho thấy hàm f là một hàm submodular và đơn điệu, điều này cho phép áp dụng thuật toán tham lam với tỉ lệ xấp xỉ bảo đảm $(1 - \frac{1}{e})$.

Trong những năm gần đây, rất nhiều hướng tiếp cận đã được đề xuất nhằm giải quyết bài toán tối đa ảnh hưởng một cách hiệu quả. Các phương pháp truyền thống bao gồm: thuật toán tham lam, lấy mẫu ngẫu nhiên theo Monte Carlo, thuật toán ước lượng nhanh, cũng như các cải tiến gần đây sử dụng quy hoạch tuyến tính ngẫu nhiên, đa tuyến tính mở rộng và các thuật toán xấp xỉ có bảo đảm lý thuyết. Khi bài toán được mở rộng với các ràng buộc chi phí, chi phí nhóm hoặc cấu trúc nó trở thành bài toán submodular với ràng buộc tương ứng.

Sở dĩ bài toán tối đa ảnh hưởng có thể giải quyết bằng công cụ tối ưu submodular là do hàm mục tiêu $f(S)$, đại diện cho kỳ vọng số lượng nút bị ảnh hưởng, là một hàm submodular. Hơn nữa, trong mô hình IC, hàm f cũng là đơn điệu vì thêm phần tử vào tập S không thể làm giảm số lượng nút bị ảnh hưởng. Từ đó, bài toán tối đa ảnh hưởng với ràng buộc lực lượng trở thành bài toán tối đa hàm submodular đơn điệu với ràng buộc lực lượng và do đó áp dụng được thuật toán tham lam chuẩn với đảm bảo hiệu quả gần tối ưu [66].

Trong khuôn khổ luận án này, một biến thể của bài toán tối đa ảnh hưởng, cụ thể là bài toán tối đa ảnh hưởng theo k chủ đề trên mạng xã hội, được lựa chọn làm trường hợp nghiên cứu thực nghiệm cho bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm.

1.5.2. Bài toán tối đa doanh thu

Bài toán tối đa doanh thu (*revenue maximization*) trong các chiến lược tiếp thị lan truyền trên mạng xã hội không chỉ nhằm mục tiêu lan tỏa thông tin đến nhiều người dùng, mà còn tối ưu doanh thu thu được từ chiến dịch đó. Khác với bài toán tối đa ảnh hưởng – vốn chỉ quan tâm đến số lượng người bị ảnh hưởng – bài toán doanh thu tính đến cả mức độ đóng góp của từng người dùng được kích hoạt, phản ánh qua mô hình doanh thu phi tuyến. Điều này cho phép mô hình hóa hiện tượng bão hòa trong hành vi tiêu dùng và tiếp nhận sản phẩm.

Bài toán được mô hình hóa dựa trên mô hình đồ thị như sau:

Định nghĩa 1.26 (Bài toán tối đa doanh thu [56])**.** Cho một mạng xã hội được biểu diễn bởi đồ thị có hướng $G = (V, E)$, trong đó mỗi cạnh $(u, v) \in E$ mang một trọng số không âm w_{uv}. Mỗi người dùng $u \in V$ có một hàm doanh thu $R_u(S)$ phụ thuộc vào tập người dùng $S \subseteq V$ đã mua hàng hoặc bị ảnh hưởng. Tổng doanh thu thu được từ tập S được tính bằng:

$$f(S) = \sum_{u \in V} R_u(S)$$

Một lựa chọn cụ thể cho $R_u(S)$ được sử dụng trong thực nghiệm là: $R_u(S) = (\sum_{v \in S} w_{uv})^{\alpha_u}$ trong đó $\alpha_u \in (0, 1)$ là tham số được chọn ngẫu nhiên cho từng người dùng và w_{uv} là cường độ ảnh hưởng từ v đến u [56]. Mục tiêu của bài toán là chọn ra tập người dùng S thỏa mãn ràng buộc sao cho tổng doanh thu thu được là lớn nhất. Mô hình này phản ánh hiệu ứng bão hòa: doanh thu tăng chậm dần khi người dùng nhận được ảnh hưởng từ nhiều nguồn.

Bài toán tối đa doanh thu theo mô hình trên là một bài toán NP-khó và không thể xấp xỉ tốt hơn $(1 - 1/e)$ trừ khi $P = NP$ [66]. Ngoài ràng buộc về lực lượng, bài toán tối đa doanh thu còn có nhiều biến thể khác như: ràng buộc chi phí, ràng buộc chi phí nhóm, tối đa doanh thu theo k-loại sản phẩm, ...

Một đặc điểm then chốt giúp giải bài toán tối đa doanh thu hiệu quả là hàm mục tiêu $f(S)$ ở trên là một hàm submodular và đơn điệu [56]. Do đó, bài toán tối đa doanh thu trở thành một bài toán tối ưu submodular. Đây là lớp bài toán đã được nghiên cứu sâu rộng với nhiều thuật toán gần đúng hiệu quả.

Trong khuôn khổ luận án này, bài toán tối đa doanh thu được triển khai thực nghiệm trong nhiều bối cảnh khác nhau nhằm khảo sát hiệu quả của các lớp thuật toán tối ưu submodular. Các biến thể được sử dụng bao gồm tối đa doanh thu với ràng buộc ngân sách, bài toán ngưỡng doanh thu dưới dạng phủ hàm submodular, tối đa doanh thu với nhiều loại sản phẩm theo mô hình k-submodular, và tối đa doanh thu cho phép đầu tư lặp lại vào cùng một người dùng theo mô hình DR-submodular.

1.5.3. Bài toán Max-Cut

Bài toán Max-Cut là một trong những bài toán cơ bản và kinh điển trong tối ưu tổ hợp, với nhiều ứng dụng trong phân cụm đồ thị, khoa học dữ liệu, vật lý lý thuyết và học máy. Bài toán nhằm phân chia tập đỉnh của một đồ thị thành hai phần sao cho tổng trọng số các cạnh nối giữa hai phần là lớn nhất. Mặc dù phát biểu đơn giản, bài toán Max-Cut thể hiện đầy đủ tính chất khó tính toán và khả năng áp dụng các kỹ thuật gần đúng có bảo đảm trong khoa học máy tính lý thuyết.

Một cách chính thức, bài toán được mô hình hóa như sau.

Định nghĩa 1.27 (Bài toán Max-Cut [65])**.** Cho một đồ thị vô hướng $G = (V, E)$, trong đó mỗi cạnh $(u, v) \in E$ mang một trọng số không âm w_{uv}. Bài toán yêu cầu tìm một tập $S \subseteq V$ sao cho hàm mục tiêu

$$f(S) = \sum_{(u,v)\in E, u\in S, v\notin S} w_{uv}$$

là lớn nhất. Đây là tổng trọng số của các cạnh có hai đầu mút nằm ở hai phía khác nhau của phép phân tách. Tập S đại diện cho một phần của phép cắt và $V \setminus S$ là phần còn lại.

Bài toán Max-Cut đã được chứng minh là NP-khó và thậm chí NP-đầy đủ, ngay cả khi các trọng số là 1 hoặc đồ thị là chính quy [46]. Do đó, việc tìm lời giải chính xác là không khả thi trong thời gian đa thức cho các trường hợp tổng quát. Tuy nhiên, nhờ đặc trưng cấu trúc của bài toán, nhiều thuật toán xấp xỉ hiệu quả đã được phát triển. Hàm $f(S)$ đo tổng trọng số các cạnh bị "cắt" bởi tập S và có thể được biểu diễn dưới dạng:

$$f(S) = \frac{1}{2} \sum_{(u,v)\in E} w_{uv} \cdot |\mathbf{1}_S(u) - \mathbf{1}_S(v)|$$

Hàm này thỏa mãn tính chất submodular do tính chất của biểu thức tuyệt đối giữa hai hàm chỉ báo $\mathbf{1}_S(u)$, trong đó hàm chỉ báo nhận giá trị 1 nếu $u \in S$ và 0 trong trường hợp ngược lại. Tuy nhiên, vì không đơn điệu nên các kỹ thuật tối ưu submodular cổ điển như tham lam không thể áp dụng trực tiếp. Thay vào đó, các thuật toán như tìm kiếm lân cận hoặc tham lam kép được sử dụng để tối ưu hàm submodular không đơn điệu.

Trong khuôn khổ luận án, bài toán Max-Cut được sử dụng làm một trường hợp thử nghiệm nhằm đánh giá hiệu quả của các thuật toán giải bài toán tối đa hàm submodular không đơn điệu với ràng buộc chi phí.

1.5.4. Bài toán tóm tắt hình ảnh

Sự bùng nổ dữ liệu hình ảnh từ các nền tảng trực tuyến như Flickr, Facebook, hay Instagram đã đặt ra một nhu cầu cấp thiết trong việc tổ chức và tóm tắt các bộ sưu tập hình ảnh cá nhân. Bài toán tóm tắt hình ảnh nhằm chọn ra một tập con đại diện từ một bộ sưu tập lớn, sao cho tập này truyền tải đầy đủ nội dung đa dạng và đặc trưng của toàn bộ tập ảnh gốc. Việc tóm tắt không chỉ hữu ích cho việc chia sẻ cá nhân mà còn có ý nghĩa trong các ứng dụng như tóm tắt video, giám sát an ninh và nhận diện cảnh. Hai yếu tố chính cấu thành một bản tóm tắt tốt là tính đại diện (fidelity) và tính đa dạng (diversity) của tập ảnh được chọn [120].

Bài toán có thể được mô hình hóa như một bài toán lựa chọn tập con với ràng buộc lực lượng như sau.

Định nghĩa 1.28 (Bài toán tóm tắt hình ảnh [90])**.** Cho tập ảnh gốc V và một giới hạn kích thước k, mục tiêu là tìm tập con $S \subseteq V, |S| \leq k$ sao cho hàm đánh giá $F(S)$ là lớn

nhất:

$$\max_{S \subseteq V, |S| \leq k} F(S)$$

với $F : 2^V \to \mathbb{R}_+$ là một hàm đánh giá chất lượng tóm tắt.

Mặc dù bài toán tối ưu hàm đánh giá tổng quát F là NP-khó, nhiều nghiên cứu thực nghiệm và lý thuyết đã chỉ ra các đặc tính mong muốn của một bản tóm tắt – bao gồm tính đại diện và tính đa dạng – có thể được mô hình hóa hiệu quả thông qua các hàm tập hợp có cấu trúc submodular. Cụ thể, các hàm submodular thể hiện rõ tính chất lợi ích biên giảm dần, tức là lợi ích thu được khi thêm một ảnh mới vào tập tóm tắt sẽ giảm dần theo kích thước của tập hiện tại. Đây là một đặc điểm phù hợp tự nhiên với mục tiêu tránh trùng lặp và tăng độ bao phủ nội dung trong tóm tắt hình ảnh. Do đó, việc giả định F là một hàm submodular đơn điệu không chỉ giúp phản ánh chính xác mục tiêu của bài toán mà còn cho phép áp dụng các thuật toán tham lam với bảo đảm xấp xỉ lý thuyết [94]. Nhờ đó, bài toán tóm tắt hình ảnh có thể được đặt trong khung bài toán tối đa hàm submodular, một lớp bài toán đã được nghiên cứu sâu rộng trong cộng đồng học thuật. Một hàm submodular cụ thể cho bài toán tóm tắt hình ảnh có thể được định nghĩa như sau:

$$f(S) = \sum_{u \in V} \max_{v \in S} w_{u,v} - \frac{1}{|V|} \sum_{u \in V} \sum_{v \in S} w_{u,v}.$$

Trong đó, $w_{u,v}$ là mức độ tương đồng giữa hai ảnh u và v, được tính dựa trên độ tương phản (*contrast*) theo công thức: $w_{u,v} = 1 - \frac{|\text{RMS}(u) - \text{RMS}(v)|}{\max(\text{RMS}(u), \text{RMS}(v))}$, với $\text{RMS}(u)$ là độ tương phản của ảnh u, được xác định bởi phương sai cường độ điểm ảnh: $\text{RMS}(u) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (I_i - \mu)^2}$, trong đó I_i là cường độ của pixel thứ i trong ảnh u, N là tổng số pixel và μ là giá trị cường độ trung bình của ảnh.

Trong khuôn khổ luận án này, nghiên cứu sinh thực nghiệm bài toán tóm tắt hình ảnh khi đánh giá hiệu quả của các thuật toán cho bài toán tối đa hàm submodular không đơn điệu với ràng buộc chi phí.

1.5.5. Bài toán đặt cảm biến

Bài toán đặt cảm biến là một bài toán quan trọng trong giám sát các hiện tượng không gian, như nhiệt độ trong một tòa nhà hoặc lượng mưa trong một khu vực rộng. Trong bối cảnh chỉ được phép triển khai một số lượng hữu hạn cảm biến, việc xác định vị trí đặt cảm biến nhằm thu thập thông tin một cách hiệu quả nhất là một vấn đề tối ưu mang tính quyết định. Thay vì dựa vào giả định hình học đơn giản như mô hình đĩa, trong đó các cảm biến có vùng phủ cố định và đều đặn, các phương pháp hiện đại ngày nay sử dụng mô hình xác suất như quá trình Gauss (Gaussian Process – GP) để mô hình hóa hiện tượng cần quan sát, từ đó dẫn đến bài toán tối ưu việc đặt cảm biến dựa trên một tiêu chí thông tin tổng quát. Bài toán được mô hình hóa như sau:

Định nghĩa 1.29 (Bài toán đặt cảm biến [70])**.** Cho một tập hợp hữu hạn các vị trí có thể đặt cảm biến V, mục tiêu là chọn ra một tập con $S \subseteq V$ gồm k vị trí sao cho hàm mục tiêu $f(S)$ đạt giá trị lớn nhất. Trong đó, $f(S)$ là một hàm mục tiêu đo lường lượng thông tin hoặc giá trị mà tập cảm biến S mang lại đối với toàn bộ hiện tượng cần quan sát. Khi đó, bài toán đặt cảm biến có thể phát biểu chính thức như sau:

$$\max_{S \subseteq V, |S|=k} f(S).$$

Trong trường hợp tiêu biểu mà $f(S)$ là hàm đo thông tin phụ thuộc lẫn nhau (mutual information) giữa tập được chọn và phần còn lại của không gian, bài toán này đã được chứng minh là NP-đầy đủ [70]. Do đó, không thể kỳ vọng một thuật toán đa thức tìm được nghiệm tối ưu trong trường hợp tổng quát.

Hiện nay, có nhiều hướng tiếp cận đã được nghiên cứu để giải bài toán này. Một cách tiếp cận cổ điển là sử dụng các tiêu chí trong thiết kế thực nghiệm như A-, D- hoặc E-optimality, nhằm tối thiểu phương sai ước lượng tham số. Tuy nhiên, các tiêu chí này thường không phản ánh trực tiếp hiệu quả dự đoán tại các vị trí không được đặt cảm biến. Một tiêu chí thay thế là entropy – chọn các vị trí có phương sai lớn nhất. Dù đơn giản, tiêu chí này có xu hướng chọn các vị trí ở biên không gian giám sát, dẫn đến lãng phí thông tin. Ngược lại, các tiêu chí như độ lợi thông tin – một trường hợp cụ thể của hàm $f(S)$ – trực tiếp tối đa khả năng dự đoán tại các vị trí chưa được đo, được chứng minh là hiệu quả hơn cả về lý thuyết lẫn thực nghiệm [70].

Trong nhiều ứng dụng thực tế, hàm mục tiêu $f(S)$ có thể được chứng minh là một hàm submodular. Tính chất này cho phép áp dụng các kết quả đã có trong bài toán tối ưu hàm submodular. Hơn nữa, các mở rộng của bài toán cho phép xử lý các ràng buộc thực tế như chi phí triển khai không đồng nhất, sự cố cảm biến, hoặc bất định trong mô hình Gaussian Process (GP). Trong các trường hợp này, hàm $f(S)$ vẫn duy trì tính chất submodular hoặc xấp xỉ submodular, cho phép sử dụng các kỹ thuật xấp xỉ hiệu quả với đảm bảo lý thuyết tương ứng [70]. Dưới đây là một số hàm submodular được sử dụng trong bài toán này:

- **Hàm đo độ hỗn loạn thông tin** (Entropy Function): Hàm này đánh giá mức độ hỗn loạn hoặc độ không chắc chắn của thông tin thu được từ tập S. Cụ thể, nếu mỗi phần tử $e \in S$ liên quan đến một biến ngẫu nhiên X_e^i, thì hàm có thể được biểu diễn dưới dạng: $f(S) = H(\bigcup_{e \in \text{supp}(S)} X_e^i)$ trong đó $H(\cdot)$ là hàm entropy đo lường lượng thông tin của tập hợp các biến ngẫu nhiên. Hàm này có tính chất submodular vì thông tin bổ sung thu được khi thêm một phần tử mới vào tập S sẽ giảm dần nếu các phần tử trước đó đã mang lại phần lớn thông tin cần thiết [98].

- **Hàm thông tin cảm biến**: Hàm này đo lường lượng thông tin thu thập được từ tập hợp các cảm biến S. Hàm này có tính chất submodular vì việc thêm một cảm biến vào

một khu vực đã được giám sát kỹ càng sẽ mang lại ít thông tin bổ sung hơn. Ví dụ:

$$f(S) = \sum_{i \in S} I(i) - \sum_{i,j \in S} R(i,j)$$

Trong đó, $I(i)$ là lượng thông tin mà cảm biến i cung cấp và $R(i,j)$ là sự trùng lặp thông tin giữa các cảm biến i và j [70].

- **Hàm bao phủ** (Coverage Function): Hàm này đo lường phạm vi bao phủ của các cảm biến trong tập S. Hàm này submodular vì thêm cảm biến mới vào khu vực đã được giám sát sẽ cung cấp ít diện tích bao phủ mới hơn. Ví dụ: $f(S) = |\cup_{i \in S} A(i)|$. Trong đó, $A(i)$ là vùng bao phủ của cảm biến i và $|\cup_{i \in S} A(i)|$ là tổng diện tích được giám sát bởi tất cả các cảm biến trong tập S [70].

- **Hàm phát hiện sự kiện** (Event Detection Function): Hàm này đo lường xác suất phát hiện sự kiện với tập hợp cảm biến S. Hàm này cũng là submodular vì việc thêm một cảm biến vào hệ thống thường có lợi ích giảm dần. Ví dụ: $f(S) = 1 - \prod_{i \in S}(1 - p(i))$ Trong đó, $p(i)$ là xác suất cảm biến i phát hiện sự kiện [82].

Bài toán đặt cảm biến tối ưu, với hàm mục tiêu có tính chất submodular, được sử dụng làm một trường hợp thực nghiệm quan trọng trong luận án. Cụ thể, các thuật toán tối ưu hàm k-submodular được áp dụng cho bài toán đặt k-loại cảm biến, nhằm minh họa tính hiệu quả về cả mặt lý thuyết và ứng dụng thực tiễn trong bài toán giám sát không gian.

1.6. Kết luận chương

Chương 1 đã giới thiệu có hệ thống các khái niệm cơ bản trong tối ưu tổ hợp, đặc biệt là các lớp hàm mục tiêu thuộc họ submodular, k-submodular và DR-submodular, cũng như các loại ràng buộc thực tiễn thường gặp như lực lượng, chi phí, chi phí nhóm. Những khái niệm và công cụ toán học này không chỉ làm rõ đặc tính cấu trúc của bài toán mà còn đóng vai trò then chốt trong việc phát triển các thuật toán xấp xỉ hiệu quả.

Mặc dù mỗi bài toán nghiên cứu trong luận án có đặc điểm riêng biệt về mục tiêu và ràng buộc, chương này đã chỉ ra rằng tất cả đều có chung một cơ sở toán học dựa trên tính chất submodular, từ đó tạo nên một khung lý thuyết thống nhất. Nhờ vậy, các chương tiếp theo có thể khai thác trực tiếp các thuộc tính đã trình bày để thiết kế và phân tích thuật toán cho từng bài toán cụ thể, bao gồm hướng tối đa hàm mục tiêu dưới ràng buộc và hướng đối ngẫu là phủ hàm mục tiêu với chi phí nhỏ nhất.

CHƯƠNG 2
BÀI TOÁN TỐI ĐA HÀM SUBMODULAR VỚI RÀNG BUỘC CHI PHÍ

Chương này tập trung vào bài toán Tối đa hàm submodular với ràng buộc chi phí (viết tắt SMK) – Bài toán nghiên cứu 1, là bài toán nền tảng trong mạch nghiên cứu của luận án. Bài toán này xét trường hợp hàm mục tiêu submodular không nhất thiết đơn điệu và nghiệm phải thỏa mãn ràng buộc ngân sách. Trên cơ sở các kết quả đã có, chương này đề xuất ba thuật toán xấp xỉ mới, mỗi thuật toán nhấn mạnh một hướng cải thiện khác nhau về độ phức tạp truy vấn, hệ số xấp xỉ hoặc khả năng tính toán song song:

- **DLA**: thuật toán tất định đạt hệ số xấp xỉ $\frac{1}{6} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$. So với SMKDETACC [53], DLA giữ cùng hệ số xấp xỉ nhưng loại bỏ sự phụ thuộc logarit vào kích thước lời giải k trong độ phức tạp truy vấn.

- **RLA**: thuật toán ngẫu nhiên đạt hệ số xấp xỉ $\frac{1}{4}-\epsilon$, tương đương với SMKRANACC [53] trong nhóm thuật toán ngẫu nhiên gần tuyến tính, đồng thời có độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$ không phụ thuộc vào k.

- **AST**: thuật toán song song đạt hệ số xấp xỉ $\frac{1}{7} - \epsilon$, cải thiện so với thuật toán song song có cùng độ phức tạp song song $O(\log n)$ đạt hệ số $\frac{1}{8} - \epsilon$ [30], đồng thời duy trì độ phức tạp truy vấn $\tilde{O}(nk)$[1].

Ba thuật toán được thiết kế cho các bối cảnh tính toán khác nhau: DLA và RLA phù hợp với môi trường xử lý tuần tự cần giảm số truy vấn oracle, trong khi AST phù hợp với môi trường song song. Về mặt kế thừa trong luận án, Chương 2 xử lý bài toán nền tảng trên hàm submodular cổ điển; các chương sau tiếp tục mở rộng hướng nghiên cứu này sang các lớp hàm tổng quát hơn và bài toán đối ngẫu. Các kết quả tương ứng với DLA, RLA và AST đã được công bố tại các hội nghị quốc tế IJCAI 2023 và IJCAI 2024.

Cấu trúc của chương được tổ chức như sau. Trước tiên, nghiên cứu sinh trình bày các khái niệm, ký hiệu cơ bản liên quan đến bài toán và các nghiên cứu liên quan đến SMK. Tiếp theo, ba thuật toán đề xuất được giới thiệu chi tiết, kèm theo các phân tích lý thuyết về độ xấp xỉ, độ phức tạp truy vấn và độ phức tạp song song. Phần đánh giá thực nghiệm sẽ trình bày hiệu quả của các thuật toán trên tập dữ liệu chuẩn, nhằm so sánh với các phương pháp hiện có trong tài liệu tham khảo. Cuối cùng là phần kết luận chương.

2.1. Mô tả bài toán

2.1.1. Định nghĩa bài toán

Hàm submodular *(Định nghĩa. 1.12)* là một lớp hàm rời rạc quan trọng trong tối ưu tổ hợp, nổi bật với tính chất lợi ích biên giảm dần: khi tập con càng lớn thì đóng góp cận biên của một phần tử mới càng nhỏ. Tuy nhiên, không phải tất cả hàm submodular đều đơn điệu. Trong trường hợp tổng quát, việc thêm phần tử vào một tập có thể làm

[1] $\tilde{O}$ được sử dụng để phân tích độ phức tạp của thuật toán khi bỏ qua các yếu tố logarit.

giảm giá trị của hàm mục tiêu. Điều này phản ánh bản chất của nhiều hệ thống thực tế, nơi các yếu tố có thể gây ra xung đột, dư thừa thông tin hoặc hiệu ứng tiêu cực khi kết hợp không hợp lý.

Động lực nghiên cứu bài toán tối đa hàm submodular với ràng buộc chi phí xuất phát từ thực tế, trong nhiều hệ thống, việc lựa chọn thêm phần tử không chỉ không cải thiện giá trị mục tiêu mà còn tiêu tốn chi phí triển khai đáng kể. Ví dụ, trong các chiến lược phân phối nội dung hoặc truyền thông xã hội, việc chọn thêm người dùng trùng lặp nhóm đối tượng có thể không mang lại thêm ảnh hưởng, trong khi vẫn tiêu hao ngân sách. Tương tự, trong học máy, việc đưa vào các đặc trưng trùng lặp hoặc mẫu dữ liệu không đa dạng có thể làm giảm hiệu quả học mà vẫn tốn chi phí thu thập hoặc xử lý. Do đó, cần thiết phải mô hình hóa các bài toán này theo hướng vừa tối ưu hiệu quả, vừa kiểm soát chi phí trong bối cảnh không đơn điệu, để phản ánh đúng các yêu cầu thực tiễn.

Một ví dụ minh họa rõ nét cho bài toán này là trong tác vụ tóm tắt văn bản tự động. Giả sử mỗi đoạn văn có một chi phí xử lý (liên quan đến độ dài, độ phức tạp ngữ nghĩa, hoặc mức độ ưu tiên) và mục tiêu là chọn ra một tập đoạn văn có tổng chi phí không vượt quá giới hạn tài nguyên cho trước. Nếu các đoạn văn được chọn có nội dung trùng lặp, bản tóm tắt trở nên dư thừa, làm giảm giá trị tổng thể của hệ thống. Do đó, cần chọn tập con đoạn văn sao cho nội dung tổng hợp là phong phú nhất, nhưng không gây lặp lại, đồng thời chi phí tổng được kiểm soát nghiêm ngặt. Cụ thể, Bài toán SMK được phát biểu chính thức như sau:

Định nghĩa 2.1 (Bài toán Tối đa hàm submodular với ràng buộc chi phí – SMK)**.** Cho một tập cơ sở có n phần tử $V = \{e_1, \ldots, e_n\}$ và một hàm mục tiêu submodular (không nhất thiết đơn điệu) $f : 2^V \mapsto \mathbb{R}_+$ dùng để đánh giá chất lượng của tập con $S \subseteq V$. Mỗi phần tử $e \in V$ được gán một chi phí dương $c(e) > 0$ và hàm chi phí $c : 2^V \mapsto \mathbb{R}_+$ là một hàm tuyến tính (modular), tức là $c(S) = \sum_{e \in S} c(e)$ và $c(S) = 0$ khi và chỉ khi $S = \emptyset$. Bài toán SMK yêu cầu tìm một tập con $S \subseteq V$ sao cho $c(S) \leq B$ nhằm tối đa giá trị $f(S)$. Một bài toán SMK được biểu diễn bởi bộ ba (f, V, B). Không mất tính tổng quát, giả định f là hàm không âm và chuẩn hóa, tức là $f(X) \geq 0$ với mọi $X \subseteq V$ và $f(\emptyset) = 0$. Ngoài ra, giả sử tồn tại một oracle có thể truy vấn giá trị $f(S)$ với một tập S bất kỳ.

Bài toán SMK có nhiều ứng dụng thực tiễn trong các hệ thống phức tạp, nơi việc lựa chọn thêm phần tử không luôn làm tăng hiệu quả và tài nguyên triển khai cần được sử dụng một cách thận trọng. Trong bài toán tóm tắt dữ liệu, chẳng hạn như tóm tắt văn bản hoặc tập ảnh, mục tiêu là chọn một tập con giàu thông tin nhưng không trùng lặp, đảm bảo bản tóm tắt đạt giá trị cao về mặt nội dung mà chi phí xử lý vẫn nằm trong giới hạn cho phép. Trong các hệ thống gợi ý nội dung, mô hình không đơn điệu giúp hạn chế việc gợi ý các nội dung tương tự nhau đến người dùng, từ đó tối đa mức độ tương tác

trong khi vẫn tuân thủ chi phí hiển thị hoặc chi phí cá nhân hóa theo từng người dùng. Trong phân tích mạng xã hội hoặc phân cụm đồ thị, bài toán này cho phép tìm các phân hoạch hoặc cắt tối ưu mà không cần giả định thêm nút hoặc cạnh luôn cải thiện độ phân biệt. Trong lĩnh vực học máy, bài toán hỗ trợ chọn đặc trưng đa dạng để tránh trùng lặp thông tin, hoặc lựa chọn mẫu huấn luyện không dư thừa trong môi trường tài nguyên tính toán bị giới hạn. Những ứng dụng này thể hiện rõ vai trò của bài toán trong các tình huống tối ưu có tính tương tác phức tạp và phi tuyến, nơi sự cân bằng giữa lợi ích và chi phí là yếu tố then chốt.

2.1.2. Nghiên cứu liên quan

Tính chất submodular và ràng buộc dạng chi phí khiến bài toán này trở nên NP-khó [43], thúc đẩy nhiều hướng tiếp cận xấp xỉ với các tiêu chí đánh giá dựa trên hệ số xấp xỉ, độ phức tạp truy vấn và khả năng tính toán song song. Phần này trình bày tổng quan các công trình nghiên cứu tiêu biểu cho bài toán SMK, bao gồm các thuật toán ngẫu nhiên, tất định, cũng như các mô hình tính toán song song và thuật toán luồng.

Các thuật toán tất định. Công trình của Gupta và cộng sự [50] là công trình đầu tiên đạt hệ số $\frac{1}{6}$ với $O(n^5)$ truy vấn bằng cách kết hợp thuật toán của Sviridenko [116] với giải pháp cho bài toán không ràng buộc. Sau đó, nhiều nỗ lực đã được thực hiện để giảm số lượng truy vấn. Thuật toán FANTOM [90] đạt hệ số $\frac{1}{10}$ trong $O(\frac{n^2}{\epsilon}\log(n))$ truy vấn. Công trình của Li [85] đưa ra thuật toán đạt hệ số $\frac{1}{9.5} - \epsilon$ với độ phức tạp $O(nk\max\{\frac{1}{\epsilon}, \log\log n\})$ cho nhiều lớp ràng buộc. Trong mô hình thuật toán luồng, Cui và cộng sự [29] đề xuất thuật toán với hệ số $\frac{1}{2.05+\rho_{\text{Alg}}}$, độ phức tạp phụ thuộc vào thuật toán offline Alg và số phần tử k. Ngoài ra, Han và cộng sự [53] cũng trình bày một thuật toán tất định khác với hệ số $\frac{1}{6} - \epsilon$ trong $O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$ truy vấn. Thuật toán có hệ số tốt nhất hiện nay là $\frac{1}{4} - \epsilon$ [114], tuy nhiên cần $O(\frac{n^3}{\epsilon}\log(\frac{n}{\epsilon}))$ truy vấn.

Các thuật toán ngẫu nhiên. Công trình đầu tiên được đề xuất bởi Lee và cộng sự [80] đạt hệ số xấp xỉ $\frac{1}{5} - \epsilon$, sau đó được cải thiện còn $\frac{1}{4} - \epsilon$ bởi Kulik và cộng sự [76]. Nhiều nghiên cứu tiếp theo đã nâng cao hiệu quả xấp xỉ với mức $\frac{1}{e} - \epsilon$ [14, 19, 35, 44]. Công trình của Buchbinder và Feldman [14] đạt hệ số tốt nhất hiện tại là $0.385 - \epsilon$ bằng cách sử dụng mở rộng đa tuyến và kỹ thuật làm tròn. Tuy nhiên, phương pháp này có độ phức tạp truy vấn rất cao. Để cải thiện hiệu năng, Amanatidis và cộng sự [3] đã đề xuất thuật toán SampleGreedy với hệ số $\frac{1}{5.83} - \epsilon$ và $O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))$ truy vấn. Sau đó, Amanatidis và cộng sự [2] tiếp tục phát triển thuật toán song song với hệ số $\frac{1}{9.465} - \epsilon$ trong $O(\frac{n^2}{\epsilon^3}\log^2(n)\log(\frac{1}{\epsilon}))$ truy vấn. Gần đây, Han và cộng sự [53] giới thiệu thuật toán ngẫu nhiên có tốc độ nhanh nhất với hệ số $\frac{1}{4} - \epsilon$ và $O(\frac{1}{\epsilon}\log(\frac{k}{\epsilon}))$ truy vấn.

Các thuật toán song song. Trong mô hình tính toán song song, khái niệm độ phức tạp song song (Định nghĩa 1.10) được giới thiệu bởi Balkanski và Singer [7], đại diện cho số vòng tuần tự cần thiết nếu các truy vấn có thể thực hiện song song. Trong bài

toán SMK, Ene và Nguyen [36] trình bày thuật toán với hệ số $\frac{1}{e} - \epsilon$ và độ phức tạp song song $O(\log^2 n)$, nhưng có độ phức tạp truy vấn cao do sử dụng đa tuyến tính mở rộng và "gradient". Sau đó, Amanatidis và cộng sự [2] cải thiện độ phức tạp song song xuống $O(\log n)$ với hệ số $\frac{1}{9.465} - \epsilon$. Gần đây, Cui và cộng sự [30] đưa ra một thuật toán song song hiệu quả đạt hệ số $\frac{1}{8} - \epsilon$ với độ phức tạp song song $O(\log n)$ và truy vấn $\tilde{O}(nk)$. Họ cũng cung cấp phiên bản cải tiến đạt hệ số $\frac{1}{7.83} - \epsilon$ nhưng cần độ phức tạp song song $O(\log^2 n)$ và hiện tại là kết quả tốt nhất trong nhóm thuật toán song song hiệu quả.

Các thuật toán luồng. Mô hình thuật toán luồng cũng được quan tâm nhằm giải quyết các bài toán tối ưu submodular trên dữ liệu lớn. Iwata và cộng sự [60], Huang và cộng sự [58] đề xuất hai thuật toán một lượt và hai lượt với hệ số xấp xỉ $\frac{1}{2.5} - \epsilon$ và truy vấn $O(\frac{n}{\epsilon^4} \log B)$. Huang và cộng sự [59] cải thiện hệ số xuống $\frac{1}{2} - \epsilon$ nhưng cần số lượt và truy vấn lớn hơn. Đặc biệt, thuật toán của Li và cộng sự [86] đạt hệ số $\frac{1}{2} - \epsilon$ chỉ với $O(\frac{n}{\epsilon} \log(\frac{1}{\epsilon}))$ truy vấn, đồng thời chứng minh không tồn tại thuật toán nào đạt hệ số hằng trong $O(\frac{n}{\log n})$ truy vấn.

Bên cạnh đó, các công trình của bài toán Tối đa hàm submodular với ràng buộc lực lượng (một trường hợp đặc biệt của bài toán SMK – khi chi phí của các phần tử đều bằng một) cũng đóng vai trò tham khảo quan trọng. Lee và cộng sự [80] đề xuất phương pháp "local search" với hệ số $\frac{1}{4}$ và $O(n^4 \log n)$ truy vấn. Gupta và cộng sự [50] cải tiến với thuật toán tham lam lặp đạt hệ số $\frac{1}{6}$ và $O(nk)$ truy vấn. [72] đạt hệ số $\frac{1}{4} - \epsilon$ với $O(n \log k)$ truy vấn. Trong hướng ngẫu nhiên, Buchbinder và cộng sự [15] sử dụng tham lam ngẫu nhiên đạt hệ số $\frac{1}{e}$ với $O(nk)$ truy vấn, sau đó được khử ngẫu nhiên bởi [13] và tăng tốc bởi [16]. Hệ số tốt nhất hiện tại vẫn là $\frac{1}{2.6}$ [14] bằng phương pháp mở rộng đa tuyến. Trong các nghiên cứu song song, Kuhnle [73], Fahrbach và cộng sự [41] đạt độ phức tạp song song $O(\log n)$ với hệ số $\frac{1}{25.64} - \epsilon$ và $\frac{1}{6} - \epsilon$. Tuy nhiên, Chen và cộng sự [23] chỉ ra lỗi trong phương pháp lấy mẫu theo ngưỡng(threshold sampling) của hai công trình trên và khắc phục bằng cách tái lập hệ số $\frac{1}{6} - \epsilon$ trong cùng độ phức tạp song song.

Các nghiên cứu hiện tại về bài toán SMK đã đề xuất nhiều thuật toán với hệ số xấp xỉ hằng số cùng độ phức tạp truy vấn và độ phức tạp song song khác nhau, phù hợp với các ứng dụng trong môi trường dữ liệu lớn hoặc yêu cầu tính toán song song. Bảng 2.1 tổng hợp chi tiết ba hướng tiếp cận chính: tất định, ngẫu nhiên và song song. Trong nhóm thuật toán tất định, thuật toán SMKDETACC hiện đạt cân bằng tốt nhất giữa hệ số xấp xỉ $\frac{1}{6} - \epsilon$ và độ phức tạp truy vấn $O(n \log k)$. Tương tự, trong nhóm ngẫu nhiên, thuật toán SMKRANACC đạt hệ số xấp xỉ $\frac{1}{4} - \epsilon$ với cùng độ phức tạp truy vấn. Với các thuật toán song song, ParSKP2 là đại diện tiêu biểu khi duy trì độ phức tạp song song $O(\log^2 n)$, độ phức tạp truy vấn $\tilde{O}(nk)$, trong khi duy trì hệ số xấp xỉ $\frac{1}{7.83} - \epsilon$.

Từ các kết quả hiện tại, luận án đặt ra ba câu hỏi nghiên cứu cụ thể cho bài toán SMK như sau:

Bảng 2.1: Bảng so sánh các thuật toán cho bài toán SMK.

Thuật toán	**Hệ số xấp xỉ**	**ĐPT Song song**	**ĐPT truy vấn**	**Phân loại**
SMKDETACC [53]	$\frac{1}{6}-\epsilon$	–	$O(n\log k)$	Tất định
SMKSTREAM [53]	$\frac{1}{6}-\epsilon$	–	$O(\frac{n}{\epsilon}\log B)$	Tất định, luồng
DLA (Thuật toán 2)	$\frac{1}{6}-\epsilon$	–	$\boldsymbol{O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))}$	Tất định
Alg.1 trong [14]	$\frac{1}{2.6}$	–	$poly(n)$	Tất định
FANTOM [90]	$\frac{1}{10}-\epsilon$	–	$O(\frac{n^2}{\epsilon}\log(n))$	Ngẫu nhiên
SAMPLEGREEDY[3]	$\frac{1}{5.83}-\epsilon$	–	$O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))$	Ngẫu nhiên
SMKRANACC [53]	$\frac{1}{4}-\epsilon$	–	$O(n\log k)$	Ngẫu nhiên
RLA (Thuật toán 4)	$\frac{1}{4}-\epsilon$	–	$\boldsymbol{O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))}$	Ngẫu nhiên
Alg.4 trong [38]	$\frac{1}{e}-\epsilon$	$O(\log^2 n)$	$\tilde{O}(n^2)$	Song song
ParKnapsack [2]	$\frac{1}{9.465}-\epsilon$	$\mathbf{O}(\log \mathbf{n})$	$\tilde{O}(n^2)$	Song song
Alg.3 trong [30]	$\frac{1}{8}-\epsilon$	$\mathbf{O}(\log \mathbf{n})$	$\tilde{O}(nk)$	Song song
ParSKP2 [30]	$\frac{1}{7.83}-\epsilon$	$O(\log^2 n)$	$\tilde{O}(nk)$	Song song
AST (Thuật toán 7)	$\boldsymbol{\frac{1}{7}-\epsilon}$	$\mathbf{O}(\log \mathbf{n})$	$\tilde{O}(nk)$	Song song

- **Câu hỏi nghiên cứu 1:** Liệu có thể thiết kế một thuật toán tất định mới cho bài toán SMK giữ được hệ số xấp xỉ hằng số $\frac{1}{6}-\epsilon$, đồng thời giảm sự phụ thuộc vào kích thước lời giải k trong độ phức tạp truy vấn?

- **Câu hỏi nghiên cứu 2:** Liệu có thể đề xuất một thuật toán ngẫu nhiên mới cho bài toán SMK giữ được hệ số xấp xỉ $\frac{1}{4}-\epsilon$ theo kỳ vọng, đồng thời loại bỏ phụ thuộc logarit vào k trong độ phức tạp truy vấn?

- **Câu hỏi nghiên cứu 3:** Liệu có thể xây dựng một thuật toán song song cho bài toán SMK giữ độ phức tạp song song $O(\log n)$, độ phức tạp truy vấn gần $\tilde{O}(nk)$, đồng thời cải thiện hệ số xấp xỉ so với các thuật toán song song cùng mức $O(\log n)$?

2.2. Thuật toán tất định

Trong hướng tiếp cận thuật toán tất định, luận án đề xuất thuật toán DLA cho bài toán SMK. Thuật toán này đạt tỉ lệ xấp xỉ $\frac{1}{6}-\epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$. So với thuật toán SMKDETACC [53], DLA giữ cùng hệ số xấp xỉ nhưng loại bỏ sự phụ thuộc logarit vào kích thước lời giải k trong độ phức tạp truy vấn. DLA được xây dựng dựa trên thuật toán nền tảng LA, một thủ tục tất định đơn giản đạt hệ số xấp xỉ hằng số $\frac{1}{19}$ với số truy vấn tuyến tính. LA chia tập cơ sở thành hai phần theo chi phí, sau đó xây dựng hai tập rời nhau bằng chiến lược tham lam theo mật độ lợi ích biên. Trên cơ sở đó,

DLA thay ngưỡng cố định bằng một dãy ngưỡng giảm dần, đồng thời xét thêm các tiền tố của hai tập ứng viên để cải thiện hệ số xấp xỉ. Phần tiếp theo trình bày LA, DLA và các phân tích lý thuyết tương ứng.

2.2.1. Thuật toán LA

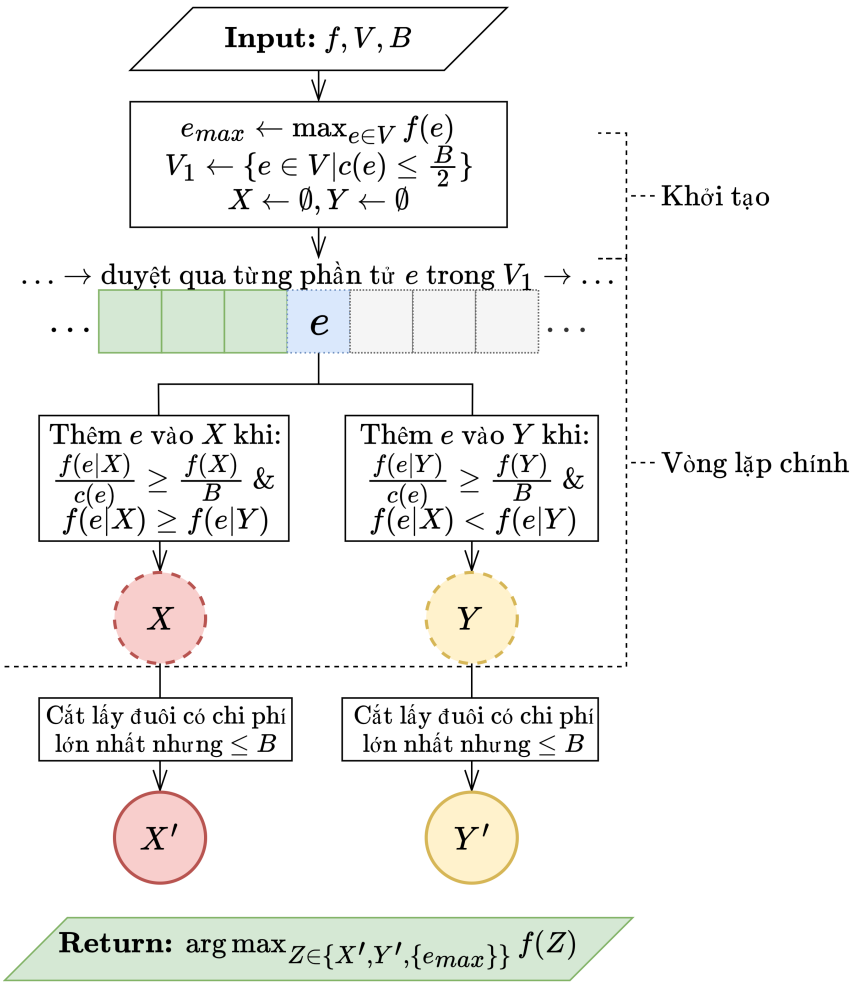
Thuật toán LA (Algorithm 1) chia tập cơ sở V thành hai tập con V_1 và V_2. Tập V_1 bao gồm các phần tử có chi phí không vượt quá $\frac{B}{2}$, trong khi V_2 bao gồm các phần tử còn lại. Chiến lược chính của LA là phân chia tập cơ sở để nhanh chóng ước lượng cận trên của nghiệm tối ưu bằng số truy vấn tuyến tính, sau đó chọn các phần tử tiềm năng vào hai tập con để đảm bảo hệ số xấp xỉ hằng số cho bài toán SMK.

Do mỗi phần tử trong V_2 có chi phí lớn hơn $\frac{B}{2}$, mọi nghiệm khả thi chỉ có thể chứa nhiều nhất một phần tử của V_2. Vì vậy, đối với phần này, LA chỉ cần xét phần tử đơn tốt nhất $e_{max} = \arg\max_{e \in V} f(e)$. Đối với V_1, thuật toán xây dựng hai tập rời nhau X và Y bằng quy tắc tham lam theo mật độ lợi ích biên. Một phần tử e được thêm vào tập $Z \in \{X, Y\}$ nếu $\frac{f(e|Z)}{c(e)}$ không nhỏ hơn ngưỡng $\frac{f(Z)}{B}$ và việc lựa chọn này vẫn giữ được tính rời nhau giữa hai tập. Bước này tạo ra hai tập X, Y phục vụ phân tích, trong đó có thể chứng minh $f(O_1) \leq 3(f(X) + f(Y))$, nhưng X và Y chưa nhất thiết là nghiệm khả thi do tổng chi phí có thể vượt quá B. Vì vậy, bước thứ hai của thuật toán lấy các đoạn cuối X' và Y' của hai dãy phần tử đã chọn sao cho chi phí không vượt quá B, rồi trả về tập có giá trị lớn nhất trong $\{X', Y', \{e_{max}\}\}$. Cụ thể, các bước được trình bày trong Thuật toán 1 và được minh họa trong Hình 2.1.

Không giống như phương pháp của Li và cộng sự [86], vốn chỉ áp dụng cho hàm mục tiêu đơn điệu và không cung cấp nghiệm khả thi, thuật toán LA xử lý được cả hàm không đơn điệu bằng cách duy trì tính rời nhau của X và Y, kết hợp với bất đẳng thức $f(O_1) \leq f(X \cup O_1) + f(Y \cup O_1)$, và từ đó suy ra $f(O) \leq f(O_1) + f(O_2)$, trong đó O là nghiệm tối ưu của bài toán, O_1 và O_2 là nghiệm tối ưu tương ứng trên V_1 và V_2.

Một ưu điểm khác của LA là có thể sử dụng giá trị $f(e_{max})$ để thiết kế và phân tích cận lý thuyết cho các thuật toán được đề xuất sau này. Trong Thuật toán 1, nếu $T = \{t_1, t_2, \ldots, t_m\}$ là thứ tự các phần tử được thêm vào $T \in \{X, Y\}$, thì $T(j) = \{t_{m-j+1}, \ldots, t_m\}$ là đoạn cuối gồm j phần tử của T. Do đó, X' và Y' là các đoạn cuối có tổng chi phí lớn nhất nhưng vẫn không vượt quá ngân sách B. Cách cắt này chuyển hai tập phân tích X, Y thành hai nghiệm khả thi X', Y'.

Phác thảo chứng minh. Phân tích của LA gồm hai ý chính. Trước hết, nhờ việc xây dựng hai tập rời nhau X và Y theo mật độ lợi ích biên, giá trị nghiệm tối ưu trên nhóm phần tử nhỏ V_1 được chặn bởi tổng giá trị của X và Y. Sau đó, vì X và Y có thể vượt ngân sách, chứng minh chỉ ra rằng các đoạn cuối khả thi X' và Y' vẫn giữ được một phần đủ lớn giá trị của X và Y; phần nghiệm tối ưu nằm trong V_2 được chặn bởi phần tử đơn tốt nhất e_{max}. Kết hợp ba thành phần này cho ra hệ số xấp xỉ của thuật toán.



Hình 2.1: Lưu đồ hoạt động của thuật toán LA

Algorithm 1 Thuật toán LA

Input: An instance (f, V, B).

Output: A feasible solution S of SMK, i.e., $c(S) \leq B$

$V_1 \leftarrow \{e \in V : c(e) \leq \frac{B}{2}\}$, $V_2 \leftarrow V \setminus V_1$, $X \leftarrow \emptyset$, $Y \leftarrow \emptyset$, $e_{max} \leftarrow \arg\max_{e \in V} f(e)$

foreach $e \in V_1$ **do**

  Find $Z \in \{X, Y\}$ such that: $Z = \arg\max_{Z \in \{X,Y\}: \frac{f(e|Z)}{c(e)} \geq \frac{f(Z)}{B}} \frac{f(e|Z)}{c(e)}$

  If exist such set Z **then** $Z \leftarrow Z \cup \{e\}$

$X' \leftarrow \arg\max_{X(j):0 \leq j \leq |X|, c(X(j)) \leq B} c(X(j))$

$Y' \leftarrow \arg\max_{Y(j):0 \leq j \leq |Y|, c(Y(j)) \leq B} c(Y(j))$, where $T(j)$ is the suffix consisting of the last j elements inserted into $T \in \{X, Y\}$ and $T(0) = \emptyset$.

$S \leftarrow \arg\max_{Z \in \{X', Y', \{e_{max}\}\}} f(Z)$

return S.

Bổ đề 2.1 cung cấp một giới hạn trên của nghiệm tối ưu trên tập V_1 thông qua hai tập rời nhau X và Y, đây là yếu tố then chốt trong việc phân tích giới hạn lý thuyết của Thuật toán 1. Bên cạnh đó, luận án sử dụng ký hiệu sau: Với $e \in X \cup Y$, ký hiệu $X^{<e}$ và $Y^{<e}$ lần lượt là tập các phần tử trong X và Y trước khi thêm e vào X hoặc Y.

Bổ đề 2.1. *Tại thời điểm kết thúc vòng lặp chính của Thuật toán 1, ta có:* $f(O_1) \leq 3(f(X) + f(Y))$.

Chứng minh. Do quy tắc lựa chọn của Thuật toán 1 và tính chất submodular của f, với mọi phần tử $e \in Y$ ta có:

$$\frac{f(e|X)}{c(e)} \leq \frac{f(e|X^{<e})}{c(e)} \leq \frac{f(e|Y^{<e})}{c(e)}. \tag{2.1}$$

Theo tính chất submodular của f, ta có:

$$\begin{aligned}
f(O_1 \cup X) - f(X) &\leq \sum_{e \in O_1 \setminus X} f(e|X) \\
&= \sum_{e \in O_1 \setminus (X \cup Y)} f(e|X) + \sum_{e \in O_1 \cap Y} f(e|X) \\
&\leq \sum_{e \in O_1 \setminus (X \cup Y)} \frac{c(e)}{B} f(X) + \sum_{e \in O_1 \cap Y} \frac{f(e|X)}{c(e)} c(e) \qquad (2.2) \\
&\leq \sum_{e \in O_1 \setminus (X \cup Y)} \frac{c(e)}{B} f(X) + \sum_{e \in O_1 \cap Y} \frac{f(e|Y^{<e})}{c(e)} c(e) \qquad (2.3) \\
&\leq f(X) + f(Y) \qquad (2.4)
\end{aligned}$$

trong đó bất đẳng thức (2.2) là do thực tế mọi phần tử $e \in O_1 \setminus (X \cup Y)$ đã không được xét để thêm vào $Z \in \{X, Y\}$ tại Dòng 3 của Thuật toán 1, tức là, $\frac{f(e|Z)}{c(e)} < \frac{f(Z)}{B}, Z \in \{X, Y\}$; bất đẳng thức (2.3) là do áp dụng (2.1).
Tương tự, ta cũng có:

$$f(O_1 \cup Y) - f(Y) \leq f(X) + f(Y). \tag{2.5}$$

Từ hai bất đẳng thức (2.4), (2.5) và do $X \cap Y = \emptyset$ ta thu được:

$$f(O_1) \leq f(O_1 \cup X) + f(O_1 \cup Y) \leq 3(f(X) + f(Y))$$

điều này hoàn tất chứng minh. □

Định lý 2.1. *Thuật toán 1 là một thuật toán tất định, đạt được tỉ lệ xấp xỉ bằng* $\frac{1}{19}$ *và yêu cầu* $O(n)$ *truy vấn.*

Chứng minh. Trước tiên, luận án chứng minh Thuật toán 1 cần nhiều nhất $3n$ truy vấn. Đầu tiên, thuật toán quét một lần qua V để tìm e_{max} (dòng 1, Thuật toán 1), tác vụ này cần n truy vấn. Sau đó, thuật toán quét một lần qua $V_1 \subseteq V$ trong vòng lặp chính (dòng 2-4). Với mỗi $e \in V_1$, nó tìm hai độ tăng biên $f(e|X) = f(X \cup \{e\}) - f(X)$ và $f(e|Y) = f(Y \cup \{e\}) - f(Y)$ và tìm $Z \in \{X, Y\}$ tại dòng 3 để thêm e vào. Ta có thể lưu trữ $f(X), f(Y)$ từ vòng lặp trước, do đó chỉ cần hai truy vấn cho tác vụ này. Vì vậy, vòng lặp chính cần $2|V_1| \leq 2n$ truy vấn và thuật toán cần nhiều nhất $3n$ truy vấn.

Tiếp theo, ta chứng minh hệ số xấp xỉ của thuật toán.

Ký hiệu $X = \{x_1, x_2, \ldots, x_{|X|}\}, X^i = \{x_1, x_2, \ldots, x_i\}, 1 \leq i \leq |X|, X^0 = \emptyset$. Giả sử $X_1 = X \setminus X' = \{x_1, \ldots, x_l\}$ và $X' = \{x_{l+1}, x_{l+2}, \ldots, x_{|X|}\}$.

Đầu tiên ta chỉ ra $f(X_1) \leq \frac{2f(X)}{3}$. Nếu $c(X) < B$, thì $X' = X$ và $X_1 = \emptyset$, mệnh đề đúng. Do đó, ta xét trường hợp $c(X) \geq B$. Do quy tắc lựa chọn phần tử để thêm vào X nên ta có:

$$f(X^j) = f(X^{j-1}) + f(x_j|X^{j-1}) \geq f(X^{j-1}) + \frac{c(x_j)}{B} f(X^{j-1}) \geq f(X^{j-1})$$

với $1 \leq j \leq |X|$. Do đó:

$$\begin{aligned} f(X) - f(X_1) &= \sum_{i=1}^{|X'|} f(x_{l+i}|X_1 \cup \{x_{l+1}, \ldots, x_{l+i-1}\}) \\ &\geq \sum_{i=1}^{|X'|} \frac{c(x_{l+i})}{B} f(X_1 \cup \{x_{l+1}, \ldots, x_{l+i-1}\}) \\ &\geq \sum_{i=1}^{|X'|} \frac{c(x_{l+i})}{B} f(X_1) = \frac{c(X')}{B} f(X_1) \\ &\geq \frac{f(X_1)}{2}. \end{aligned}$$

Bất đẳng thức cuối cùng là do mỗi phần tử trong V_1 có chi phí nhiều nhất $\frac{B}{2}$ và quy tắc lựa chọn của X' cho: $B \geq c(X') \geq B - \frac{B}{2} = \frac{B}{2}$.

Do đó, $f(X_1) \leq \frac{2f(X)}{3}$. Theo tính chất submodular của f, ta có:

$$f(X') \geq f(X) - f(X_1) \geq \frac{f(X)}{3}.$$

Tương tự, ta có: $f(Y') \geq \frac{f(Y)}{3}$. Mặt khác, nghiệm tối ưu trên V_2 có nhiều nhất một phần tử, nên $f(O_2) \leq f(e_{max})$. Cuối cùng, từ tính chất submodular, quy tắc lựa chọn của nghiệm cuối cùng trong Thuật toán 1 và Bổ đề 2.1 ta thu được:

$$\begin{aligned} f(O) &\leq f(O \cap V_1) + f(O \cap V_2) \leq f(O_1) + f(O_2) \\ &\leq 3(f(X) + f(Y)) + f(e_{max}) \\ &\leq 9(f(X') + f(Y')) + f(e_{max}) \leq 19 f(S) \end{aligned}$$

điều này hoàn tất chứng minh. □

2.2.2. Thuật toán DLA

Thuật toán DLA (Thuật toán 2) là một thuật toán xấp xỉ tất định đạt tỉ lệ $\frac{1}{6} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon} \log(\frac{1}{\epsilon}))$. Ý tưởng chính của DLA là dùng LA để có một cận ban đầu cho giá trị tối ưu, sau đó xây dựng hai tập rời nhau bằng chiến lược tham lam có ngưỡng giảm dần. Khác với LA, thuật toán không chỉ dùng hai tập cuối cùng X, Y, mà còn xét các tiền tố của chúng trong pha thứ hai. Cách này giúp thuật toán chọn được một

phần nghiệm có chi phí phù hợp với ngân sách còn lại, rồi bổ sung thêm một phần tử tốt nhất để cải thiện giá trị nghiệm.

Algorithm 2 Thuật toán DLA

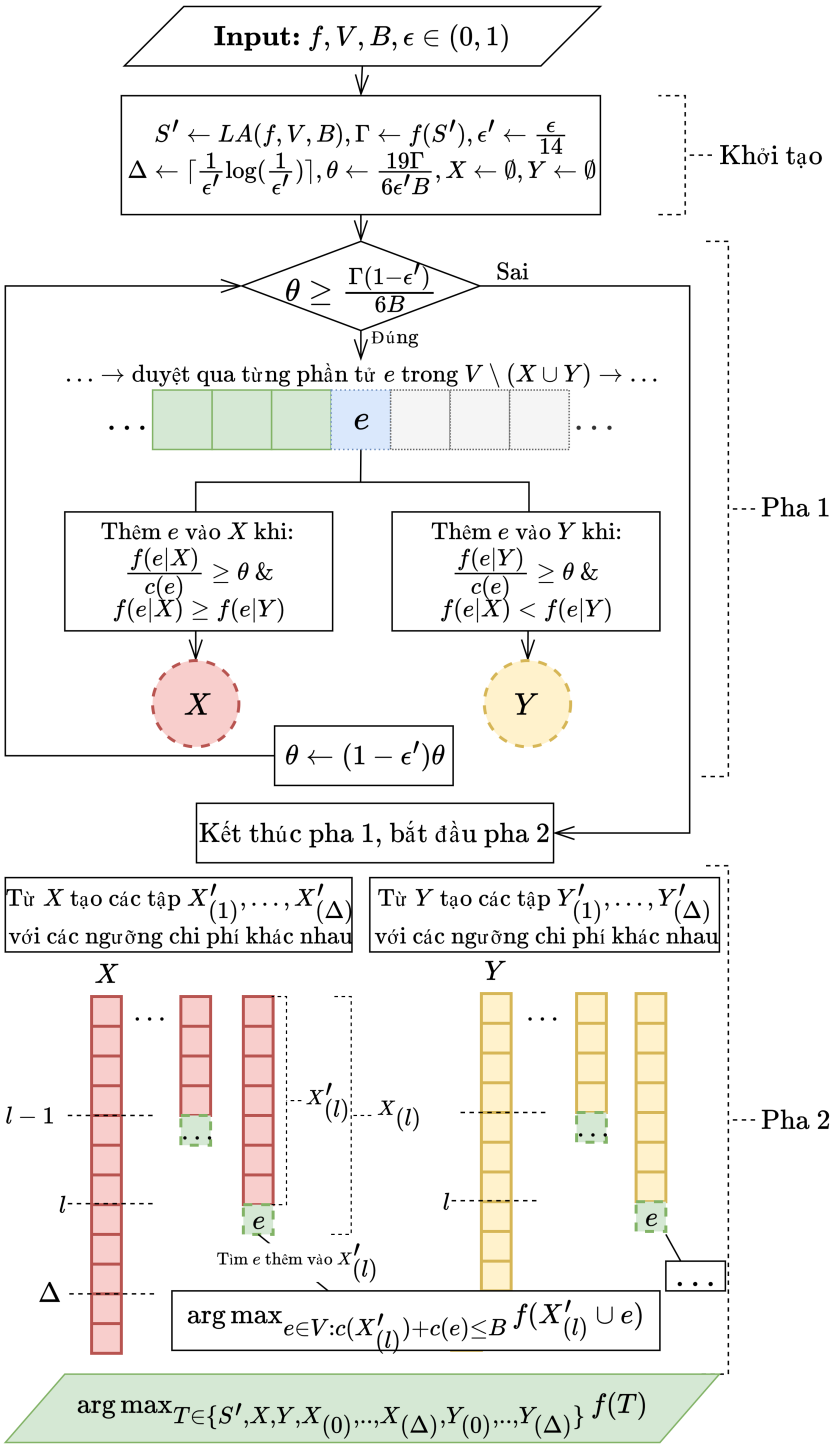
Input: An instance (f, V, B), parameters $\epsilon \in (0, 1)$
Output: A feasible solution S of SMK, i.e., $c(S) \leq B$

1: $S' \leftarrow LA(f, V, B), \Gamma \leftarrow f(S'), \epsilon' \leftarrow \frac{\epsilon}{14}$
2: $\Delta \leftarrow \lceil \frac{1}{\epsilon'} \log(\frac{1}{\epsilon'}) \rceil, \theta \leftarrow \frac{19\Gamma}{6\epsilon' B}, X \leftarrow \emptyset, Y \leftarrow \emptyset$
3: **while** $\theta \geq \frac{\Gamma(1-\epsilon')}{6B}$ **do**
4: **foreach** $e \in V \setminus (X \cup Y)$ **do**
5: Find $T \in \{X, Y\}$ such that: $c(T \cup \{e\}) \leq B$ and $T = \arg\max_{T \in \{X,Y\}, \frac{f(e|T)}{c(e)} \geq \theta} \frac{f(e|T)}{c(e)}$
6: **If** exist such set T **then** $T \leftarrow T \cup \{e\}$
7: $\theta \leftarrow (1 - \epsilon')\theta$
8: **for** $l = 0$ *to* Δ **do**
9: $X'_{(l)} \leftarrow \arg\max_{X^i : c(X^i) \leq \epsilon' B(1+\epsilon')^l, i \leq |X|} i$
10: $Y'_{(l)} \leftarrow \arg\max_{Y^i : c(Y^i) \leq \epsilon' B(1+\epsilon')^l, i \leq |Y|} i$
11: $e_X \leftarrow \arg\max_{e \in V : c(X'_{(l)} \cup \{e\}) \leq B} f(X'_{(l)} \cup \{e\})$
12: $e_Y \leftarrow \arg\max_{e \in V : c(Y'_{(l)} \cup \{e\}) \leq B} f(Y'_{(l)} \cup \{e\})$
13: $X_{(l)} \leftarrow X'_{(l)} \cup \{e_X\}, Y_{(l)} \leftarrow Y'_{(l)} \cup \{e_Y\}$
14: $S \leftarrow \arg\max_{T \in \{S', X, Y, X_{(0)}, \ldots, X_{(\Delta)}, Y_{(0)}, \ldots, Y_{(\Delta)}\}} f(T)$
15: **return** S.

Thuật toán DLA nhận đầu vào là một bài toán (f, V, B) cùng tham số ϵ và gồm ba thành phần: khởi tạo, pha 1 và pha 2. Ở bước khởi tạo, thuật toán gọi LA để thu được nghiệm ứng viên S' và đặt $\Gamma = f(S')$. Theo hệ số xấp xỉ của LA, giá trị tối ưu opt nằm trong khoảng được kiểm soát bởi Γ, nên Γ được dùng để thiết lập dãy ngưỡng cho các bước tiếp theo. Trong pha 1 (dòng 1–7), thuật toán xây dựng hai tập rời nhau X và Y. Mỗi vòng lặp quét qua các phần tử chưa được chọn; một phần tử được thêm vào tập $T \in \{X, Y\}$ nếu chi phí còn khả thi và mật độ lợi ích biên $\frac{f(e|T)}{c(e)}$ đạt ít nhất ngưỡng hiện tại θ. Ở đây, θ là mức yêu cầu tối thiểu về lợi ích biên trên một đơn vị chi phí: ngưỡng lớn giúp chọn các phần tử rất tốt ở đầu thuật toán, còn việc giảm dần θ theo hệ số $(1 - \epsilon')$ cho phép thuật toán xét thêm các phần tử có mật độ thấp hơn nhưng vẫn có ích cho nghiệm.

Pha 2 (dòng 8–13) xử lý vấn đề còn lại: các tập X, Y được tạo ở pha 1 có thể chưa phải là phần nghiệm có chi phí phù hợp nhất để kết hợp với một phần tử quan trọng của nghiệm tối ưu. Gọi T^i là tiền tố gồm i phần tử đầu tiên được thêm vào $T \in \{X, Y\}$. Trong phân tích, $r = \arg\max_{o \in O} c(o)$ là phần tử có chi phí lớn nhất trong nghiệm tối ưu O; ký hiệu này chỉ dùng cho chứng minh vì thuật toán không biết O. Nếu biết r, tiền tố tự nhiên cần xét là $T' = \arg\max_{T^i : c(T^i) \leq B - c(r)} i$, tức là tiền tố dài nhất còn đủ ngân sách để ghép với r. Do không biết $c(r)$, DLA duyệt một lưới giá trị $\epsilon' B(1 + \epsilon')^l$ để xấp xỉ các

khả năng của phần ngân sách còn lại, rồi với mỗi tiền tố thu được, thuật toán thêm phần tử khả thi tốt nhất. Cụ thể, các bước được trình bày trong Thuật toán 2 và được minh họa trong Hình 2.2.



Hình 2.2: Lưu đồ hoạt động của thuật toán DLA

Như vậy, pha 1 tạo ra hai dãy ứng viên X, Y theo mật độ lợi ích biên, còn pha 2 chọn các tiền tố có chi phí đại diện cho nhiều mức ngân sách khác nhau và thử bổ sung một phần tử tốt nhất. Hai Bổ đề sau đây phân tích chất lượng nghiệm cuối cùng trong

hai tình huống của phần tử r: $c(r) < (1-\epsilon')B$, khi phần tử lớn nhất của nghiệm tối ưu chưa chiếm gần hết ngân sách; và $c(r) \geq (1-\epsilon')B$, khi một phần tử của nghiệm tối ưu gần như quyết định phần lớn ngân sách.

Để phân tích lý thuyết cho DLA, luận án định nghĩa hoặc nhắc lại các ký hiệu trong Bảng 2.2.

Bảng 2.2: Bảng các ký tự toán học dùng trong phân tích thuật toán DLA

Ký hiệu	**Ý nghĩa**
O	Nghiệm tối ưu của bài toán SMK và $\mathsf{opt} = f(O)$
X^i	Giả sử $X = \{x_1, x_2, \dots x_{\lvert X\rvert}\}$ theo thứ tự các phần tử được thêm vào X, ký hiệu $X^i = \{x_1, x_2, \dots x_i\}$ là tiền tố gồm i phần tử đầu tiên
r	Phần tử có chi phí lớn nhất trong nghiệm tối ưu O, tức là $r = \arg\max_{o \in O} c(o)$; ký hiệu này chỉ dùng trong phân tích
t	$t = \max\{i : c(X^i) + c(r) \leq B\}$, tức là chỉ số tiền tố dài nhất của X còn có thể ghép với r mà không vượt quá ngân sách B
$X'_{(l)}, Y'_{(l)}$	Các tiền tố của X và Y được chọn ở pha 2 với ngưỡng chi phí $\epsilon' B(1+\epsilon')^l$
X_j, Y_j	Lần lượt là X và Y sau vòng lặp thứ j của vòng lặp đầu tiên trong Thuật toán 2
$X^{<e}, Y^{<e}$	Lần lượt là tập các phần tử trong X và Y trước khi thêm $e \in X \cup Y$ vào X hoặc Y
θ_i	Là giá trị θ tại vòng lặp thứ i; θ đóng vai trò ngưỡng mật độ lợi ích biên tối thiểu để một phần tử được chọn
$\theta_{(j)}$	Là giá trị θ khi phần tử x_j được thêm vào X
θ_{last}	Là giá trị θ tại vòng lặp cuối cùng của vòng lặp đầu tiên

Phác thảo chứng minh. Phân tích của DLA dựa trên việc so sánh nghiệm tối ưu O với các tập X, Y được tạo bởi dãy ngưỡng giảm dần. Phần tử có chi phí lớn nhất r trong O được dùng để chia phân tích thành hai tình huống: nếu r chưa chiếm gần hết ngân sách, pha 2 có thể tìm một tiền tố của X hoặc Y có chi phí gần với phần ngân sách còn lại để ghép với một phần tử tốt; nếu r gần như chiếm toàn bộ ngân sách, một phần tử đơn hoặc một tiền tố ngắn đã đủ tạo cận tốt. Hai bổ đề tương ứng xử lý hai tình huống này, sau đó định lý kết hợp các cận để suy ra tỉ lệ $\frac{1}{6} - \epsilon$.

Bổ đề 2.2 xét trường hợp phần tử có chi phí lớn nhất trong nghiệm tối ưu không chiếm gần hết ngân sách. Khi đó, phần ngân sách còn lại $B - c(r)$ vẫn lớn hơn $\epsilon' B$, nên một tiền tố thích hợp của X hoặc Y có thể được pha 2 của thuật toán xấp xỉ thông qua lưới giá trị $\epsilon' B(1+\epsilon')^l$.

Bổ đề 2.2. *Nếu $c(r) < (1-\epsilon')B$, một trong hai điều sau xảy ra:* ***a)*** $f(S) \geq \frac{1}{6(1+\epsilon')}\mathsf{opt}$*;* ***b)*** *Tồn tại tập con $X' \subseteq X$ sao cho*

$$f(O \cup X') \leq 2f(S) + \max\left\{\frac{1+\epsilon'}{1-\epsilon'}f(S), \frac{(1-\epsilon')}{6}\mathsf{opt}\right\}.$$

Tương tự, một trong hai điều sau xảy ra: ***c)*** $f(S) \geq \frac{1}{6(1+\epsilon')}\mathsf{opt}$*;* ***d)*** *Tồn tại tập con $Y' \subseteq Y$ sao cho*

$$f(O \cup Y') \leq 2f(S) + \max\left\{\frac{1+\epsilon'}{1-\epsilon'}f(S), \frac{(1-\epsilon')}{6}\mathsf{opt}\right\}.$$

Chứng minh. Trong trường hợp này ta có $B - c(r) > \epsilon' B$. Chứng minh dùng t để xác định tiền tố dài nhất X^t của X vẫn có thể kết hợp với r trong giới hạn ngân sách. Nếu $X^t = X$, toàn bộ tập X vẫn phù hợp với phần tử r; nếu $X^t \subset X$, tiền tố kế tiếp X^{t+1} là điểm đầu tiên vượt quá phần ngân sách còn lại, nên pha 2 sẽ chọn một tiền tố gần với mức chi phí này. Ta xét các trường hợp sau:

Trường hợp 1. Nếu $X^t = X$ sau khi kết thúc pha 1, tức là sau vòng lặp ở các dòng 1–7 của Thuật toán 2. Theo quy tắc lựa chọn của thuật toán, mỗi phần tử $e \in Y^{<x_t}$ có mật độ lợi ích biên thỏa mãn:

$$\frac{f(e|X^t)}{c(e)} \leq \frac{f(e|X^{<e})}{c(e)} \leq \frac{f(e|Y^{<e})}{c(e)}. \tag{2.6}$$

Mỗi phần tử $e \in O \setminus (X^t \cup Y^{<x_t})$ có mật độ lợi ích biên với X^t nhỏ hơn θ_{last}, tức là $\frac{f(e|X^t)}{c(e)} \leq \theta_{last}$, nên ta thu được:

$$\begin{aligned}
f(X^t \cup O) - f(X^t) &\leq \sum_{e \in O \setminus X^t} f(e|X) \\
&= \sum_{e \in O \cap Y^{<x_t}} f(e|X^t) + \sum_{e \in O \setminus (X^t \cup Y^{<x_t})} f(e|X) \\
&\leq \sum_{e \in O \cap Y^{<x_t}} \frac{f(e|X^t)}{c(e)} c(e) + \sum_{e \in O \setminus (X^t \cup Y^{<x_t})} c(e)\theta_{last} \qquad (2.7) \\
&< \sum_{e \in O \cap Y^{<x_t}} \frac{f(e|Y^{<e})}{c(e)} c(e) + c(O)\theta_{last} \qquad (2.8) \\
&\leq f(Y^{<x_t}) + \frac{(1-\epsilon')\mathsf{opt}}{6}, \qquad (2.9)
\end{aligned}$$

trong đó bất đẳng thức (2.8) là do (2.6), bất đẳng thức (2.9) là do thực tế $\frac{f(e|Y^{<e})}{c(e)} \geq \theta > 0$ với mọi $e \in Y$. Từ (2.9) và lưu ý $f(X^t) \leq f(X) \leq f(S)$ và $f(Y^{<x_t}) \leq f(Y) \leq f(S)$, ta có:

$$f(X^t \cup O) < f(X^t) + f(Y^{<x_t}) + \frac{(1-\epsilon')\mathsf{opt}}{6} \leq 2f(S) + \frac{(1-\epsilon')\mathsf{opt}}{6}.$$

Trường hợp 2. $X^t \subset X$ sau khi kết thúc pha 1. Khi đó X^{t+1} là tiền tố đầu tiên không còn ghép được với r trong ngân sách, nên cần dùng pha 2 để tìm một tiền tố có

chi phí gần với $B - c(r)$. Trong trường hợp này, X chứa ít nhất $t+1$ phần tử và $c(X^{t+1}) > B - c(r) > \epsilon' B$. Ta xét vòng lặp thứ hai của Thuật toán 2. Vì $\epsilon' B < B - c(r) \leq B$, tồn tại một số nguyên l sao cho

$$L = (1+\epsilon')^l \epsilon' B \leq B - c(r) < (1+\epsilon')^{l+1} \epsilon' B = L(1+\epsilon').$$

Giả sử $X'_{(l)} = X^i$ với một số i. Theo lựa chọn của $X'_{(l)}$ tại Dòng 11 trong Thuật toán 2 và $c(X) \geq c(X^{t+1}) > \epsilon' B$, ta có $c(X^i) \leq L < c(X^{i+1})$, do đó:

$$c(X^{i+1}) > L > \frac{B - c(r)}{1+\epsilon'} \geq \frac{\epsilon' B}{1+\epsilon'}. \tag{2.10}$$

Ta tiếp tục xét hai trường hợp con sau:

Trường hợp 2.1. Nếu phần tử thứ $i+1$ của X được thêm ngay trong vòng quét đầu tiên của pha 1, khi ngưỡng còn ở mức cao nhất θ_1, thì bản thân tiền tố X^{i+1} đã đủ lớn để suy ra cận dưới cho $f(S)$. Ta có:

$$f(S) \geq f(X^{i+1}) \geq c(X^{i+1})\theta_1 > \frac{\epsilon' B}{1+\epsilon'} \frac{19\Gamma}{6\epsilon' B} \geq \frac{\mathsf{opt}}{6(1+\epsilon')},$$

từ đó suy ra Bổ đề đúng.

Trường hợp 2.2. Nếu phần tử thứ $i+1$ của X chỉ được thêm ở một vòng quét sau của pha 1, tức là tại vòng $j \geq 2$, thì các phần tử chưa được chọn ở vòng trước đều có mật độ lợi ích biên nhỏ hơn ngưỡng tương ứng. Nhận xét này cho phép chặn phần đóng góp còn lại của O bằng giá trị của Y và tiền tố X^{i+1}. Với mọi phần tử $e \in V \setminus (X^i \cup Y^{<x_i})$, mật độ lợi ích biên của nó với X^i nhỏ hơn ngưỡng tại vòng lặp trước (trong vòng lặp đầu tiên), tức là:

$$\frac{f(e|X^i)}{c(e)} < \frac{\theta_{(i+1)}}{1-\epsilon'}. \tag{2.11}$$

Mặt khác, mật độ lợi ích biên của mỗi phần tử trong X^{i+1} lớn hơn hoặc bằng ngưỡng $\theta_{(i+1)}$, nên ta có:

$$\frac{f(X^{i+1})}{c(X^{i+1})} = \frac{\sum_{k=1}^{i+1} f(x_k|X^{k-1})}{c(X^{i+1})} \geq \frac{\sum_{k=1}^{i+1} \theta_{(i+1)} c(x_k)}{c(X^{i+1})} = \theta_{(i+1)}. \tag{2.12}$$

Ký hiệu $O_1 = O \cap Y^{<x_i}$ và $O_2 = O \setminus (X^i \cup O_1)$. Kết hợp các bất đẳng thức (2.10), (2.11) và (2.12), ta thu được:

$$\begin{aligned}
f(X^i \cup O) - f(X^i \cup \{r\}) &\leq \sum_{e \in O \setminus X^i} f(e|X^i \cup \{r\}) \leq \sum_{e \in O \setminus (X^i \cup \{r\})} f(e|X^i) \\
&= \sum_{e \in O_1 \setminus \{r\}} f(e|X^i) + \sum_{e \in O_2 \setminus \{r\}} f(e|X^i) \\
&< \sum_{e \in O_1 \setminus \{r\}} f(e|Y^{<e}) + \sum_{e \in O_2 \setminus \{r\}} c(e) \frac{\theta_{(i+1)}}{1-\epsilon'} \text{ (do (2.11))} \\
&\leq f(Y) + (B - c(r)) \frac{\theta_{(i+1)}}{1-\epsilon'}
\end{aligned}$$

$$\leq f(Y) + \frac{(B - c(r))f(X^{i+1})}{(1-\epsilon')c(X^{i+1})} \text{ (do (2.12))}$$
$$< f(Y) + \frac{1+\epsilon'}{1-\epsilon'} f(X^{i+1}) \text{ (do (2.10))}$$

từ đó suy ra: $f(X^i \cup O) < f(X^i \cup \{r\}) + f(Y) + \frac{1+\epsilon'}{1-\epsilon'} f(X^{i+1})$.

Theo quy tắc lựa chọn của e_X tại dòng 11 trong Thuật toán 2, $f(X^i \cup \{r\}) \leq f(X^i \cup \{e_X\}) \leq f(S)$, do đó ta có: $f(X^i \cup O) \leq 2f(S) + \frac{1+\epsilon'}{1-\epsilon'} f(S)$.

Kết hợp cả hai trường hợp, ta có kết quả trong Bổ đề. Theo lập luận tương tự, ta cũng thu được kết quả tương tự đối với một tập con $Y' \subseteq Y$. □

Bổ đề 2.3. *Nếu $c(r) \geq (1-\epsilon')B$, một trong hai điều sau xảy ra:* ***e)*** $f(S) \geq \frac{(1-\epsilon')^2}{6}\mathsf{opt}$*;* ***f)*** *Tồn tại tập con $X' \subseteq X$ sao cho*

$$f(O \cup X') \leq 2f(S) + \max\left\{\frac{(1-\epsilon')\mathsf{opt}}{6}, f(S) + \frac{\epsilon'\mathsf{opt}}{6}\right\}.$$

Tương tự, một trong hai điều sau xảy ra: ***g)*** $f(S) \geq \frac{(1-\epsilon')^2}{6}\mathsf{opt}$*;* ***h)*** *Tồn tại tập con $Y' \subseteq Y$ sao cho*

$$f(O \cup Y') \leq 2f(S) + \max\left\{\frac{(1-\epsilon')\mathsf{opt}}{6}, f(S) + \frac{\epsilon'\mathsf{opt}}{6}\right\}.$$

Chứng minh. Trong trường hợp này, ta có $c(O \setminus \{r\}) \leq \epsilon' B, c(X^t) \leq \epsilon' B$.

Trường hợp 1. Nếu X^t là X sau khi kết thúc vòng lặp đầu tiên. Bằng phép biến đổi tương tự từ (2.7) đến (2.9) trong chứng minh Bổ đề 2.2, ta cũng thu được:

$$f(X^t \cup O) < 2f(S) + \frac{(1-\epsilon')\mathsf{opt}}{6}.$$

Trường hợp 2. Nếu $X^t \subset X$, thì X chứa ít nhất $t+1$ phần tử. Xét vòng lặp đầu tiên của thuật toán, $\theta_j = \frac{19\Gamma(1-\epsilon')^j}{6\epsilon' B} \in [\frac{(1-\epsilon')\mathsf{opt}}{6B}, \frac{19\mathsf{opt}}{6\epsilon' B}]$ với mọi vòng lặp j. Vì ngưỡng θ giảm dần theo hệ số $1-\epsilon'$ sau mỗi vòng lặp, nên tồn tại một vòng lặp j thỏa mãn:

$$\frac{(1-\epsilon')\mathsf{opt}}{6B} \leq \theta_j = \frac{19\Gamma(1-\epsilon')^j}{6\epsilon' B} < \frac{\mathsf{opt}}{6B}.$$

Ta tiếp tục xét hai trường hợp con sau:

- Nếu $X^{t+1} \subseteq X_j$. Nếu $c(X_j) \geq (1-\epsilon')B$, thì

$$f(S) \geq f(X_j) \geq c(X_j)\theta_j \geq \frac{(1-\epsilon')^2}{6}\mathsf{opt}.$$

Nếu $c(X_j) < (1-\epsilon')B$. Ký hiệu $O_1 = O \cap Y_j$ và $O_2 = O \setminus (X_j \cup O_1)$. Vì $c(X_j) + \max_{e \in O\setminus\{r\}} c(e) \leq c(X_j) + c(O \setminus \{r\}) < B$, ta có $\frac{f(e|X_j)}{c(e)} < \theta_j$ với mọi $e \in O_2 \setminus \{r\}$. Do đó:

$$f(X_j \cup O) - f(X_j \cup \{r\}) \leq \sum_{e \in O\setminus\{r\}} f(e|X_j)$$
$$= \sum_{e \in O_1\setminus\{r\}} f(e|X_j) + \sum_{e \in O_2\setminus\{r\}} f(e|X_j)$$

$$< \sum_{e \in O_1 \setminus \{r\}} f(e|Y^{<e}) + \sum_{e \in O_2 \setminus \{r\}} c(e)\theta_j$$
$$\leq f(Y) + \epsilon' B \frac{\mathsf{opt}}{6B} = f(Y) + \frac{\epsilon'}{6}\mathsf{opt}.$$

Theo quy tắc lựa chọn của nghiệm cuối cùng và lưu ý $f(r) \leq f(e_{max}) \leq f(S') \leq f(S)$, ta thu được:

$$f(X_j \cup O) \leq f(X_j \cup \{r\}) + f(Y) + \frac{\epsilon'\mathsf{opt}}{6}$$
$$\leq f(X_j) + f(r) + f(Y) + \frac{\epsilon'}{6}\mathsf{opt}$$
$$\leq 3f(S) + \frac{\epsilon'}{6}\mathsf{opt}.$$

- Nếu $X_j \subset X^{t+1}$. Với mọi phần tử $e \in V \setminus (X^t \cup Y^{<x_t})$, mật độ lợi ích biên của nó đối với X^t nhỏ hơn ngưỡng tại vòng lặp trước (trong vòng lặp đầu tiên), do đó:

$$\frac{f(e|X^t)}{c(e)} < \frac{\theta_{(t+1)}}{1-\epsilon'} \leq \theta_j < \frac{\mathsf{opt}}{6B}.$$

Ký hiệu $O_1 = O \cap Y^{<x_t}$ và $O_2 = O \setminus (X^t \cup O_1)$. Với lưu ý $c(O_2 \setminus \{r\}) < \epsilon' B$, ta có:

$$f(X^t \cup O) - f(X^t \cup \{r\}) \leq \sum_{e \in O \setminus \{r\}} f(e|X^t \cup \{r\}) \leq \sum_{e \in O \setminus \{r\}} f(e|X^t)$$
$$= \sum_{e \in O_1 \setminus \{r\}} f(e|X^t) + \sum_{e \in O_2 \setminus \{r\}} f(e|X^t)$$
$$< \sum_{e \in O_1 \setminus \{r\}} f(e|Y^{<e}) + \sum_{e \in O_2 \setminus \{r\}} c(e)\theta_j$$
$$\leq f(Y^{<x_t}) + c(O_2 \setminus \{r\})\frac{\mathsf{opt}}{6B}$$
$$\leq f(Y) + \frac{\epsilon'\mathsf{opt}}{6}.$$

Điều này suy ra:

$$f(X^t \cup O) \leq f(X^t \cup \{r\}) + f(Y) + \frac{\epsilon'}{6}\mathsf{opt}$$
$$\leq f(X^t) + f(r) + f(Y) + \frac{\epsilon'}{6}\mathsf{opt}$$
$$\leq 3f(S) + \frac{\epsilon'}{6}\mathsf{opt}.$$

Kết hợp cả hai trường hợp, ta có kết quả trong Bổ đề. Theo lập luận tương tự, ta cũng có kết quả tương tự với $Y' \subseteq Y$. □

Định lý 2.2. *Với mọi $\epsilon \in (0,1)$, thuật toán DLA là thuật toán tất định có độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$ và đạt được tỉ lệ xấp xỉ $\frac{1}{6} - \epsilon$.*

Chứng minh. Độ phức tạp truy vấn của Thuật toán 2 được tính bằng cách kết hợp thao tác của Thuật toán 1 và hai vòng lặp chính của Thuật toán 2. Vòng lặp thứ nhất và thứ

hai lần lượt có nhiều nhất $\lceil \frac{1}{\epsilon'} \log(\frac{19}{\epsilon'}) \rceil + 1$ và $\lceil \frac{1}{\epsilon'} \log(\frac{1}{\epsilon'}) \rceil$ vòng lặp. Mỗi vòng lặp trong hai vòng này cần $O(n)$ truy vấn; do đó ta thu được tổng số truy vấn nhiều nhất là:

$$3n + n(\lceil \frac{1}{\epsilon'} \log(\frac{19}{\epsilon'}) \rceil + 1) + n\lceil \frac{1}{\epsilon'} \log(\frac{1}{\epsilon'}) \rceil = O(\frac{n}{\epsilon} \log(\frac{1}{\epsilon})).$$

Để chứng minh hệ số xấp xỉ, luận án xét hai trường hợp sau:

Trường hợp 1. Nếu $c(r) \geq (1-\epsilon')B$. Bằng cách sử dụng Bổ đề 2.3, ta xét hai tình huống: nếu xảy ra **e)** hoặc **g)**. Vì $\epsilon' = \frac{\epsilon}{14} < \frac{1}{14}$, ta có: $\mathsf{opt} \leq \frac{6f(S)}{(1-\epsilon')^2} \leq 6(1+\frac{14}{13}\epsilon')^2 f(S) < (6+\epsilon)f(S)$, do đó Định lý được chứng minh. Ta xét trường hợp ngược lại: cả **e)** và **h)** đều xảy ra. Khi đó tồn tại $X' \subseteq X$, $Y' \subseteq Y$ và $X' \cap Y' = \emptyset$ thỏa mãn:

$$\begin{aligned} \mathsf{opt} = f(O) &\leq f(O \cup X') + f(O \cup Y') \\ &\leq 4f(S) + 2\max\{(1-\epsilon')\frac{\mathsf{opt}}{6}, f(S) + \epsilon'\frac{\mathsf{opt}}{6}\}. \end{aligned} \tag{2.13}$$

Ta xét hai trường hợp con: Nếu $f(S) \geq \frac{\mathsf{opt}}{6}$, Định lý đúng. Nếu $f(S) < \frac{\mathsf{opt}}{6}$, thay vào (2.13) ta được:

$$\mathsf{opt} < 4f(S) + \frac{1+\epsilon'}{3}\mathsf{opt} \Rightarrow \mathsf{opt} \leq \frac{12f(S)}{2-\epsilon'} < (6+\epsilon)f(S).$$

Trường hợp 2. Nếu $c(r) < (1-\epsilon')B$. Áp dụng Bổ đề 2.2, ta xét hai tình huống: nếu xảy ra **a)** hoặc **c)**, ta có $f(S) \geq \frac{\mathsf{opt}}{6(1+\epsilon')} \Rightarrow \mathsf{opt} \leq (6+6\epsilon')f(S)$ và Định lý đúng. Nếu cả **b)** và **d)** đều xảy ra. Khi đó tồn tại $X' \subseteq X$, $Y' \subseteq Y$ và $X' \cap Y' = \emptyset$ thỏa mãn:

$$\begin{aligned} \mathsf{opt} = f(O) &\leq f(O \cup X') + f(O \cup Y') \\ &\leq 4f(S) + 2\max\{\frac{1+\epsilon'}{1-\epsilon'}f(S), \frac{(1-\epsilon')}{6}\mathsf{opt}\}. \end{aligned} \tag{2.14}$$

Nếu $f(S) \geq \frac{\mathsf{opt}}{6}$, Định lý đúng. Ta xét trường hợp $f(S) < \frac{\mathsf{opt}}{6}$, thay vào (2.14) ta được: $\mathsf{opt} < 4f(S) + \frac{1+\epsilon'}{1-\epsilon'}\frac{\mathsf{opt}}{3}$. Suy ra:

$$\mathsf{opt} < \frac{6(1-\epsilon')}{1-2\epsilon'}f(S) = (6 + \frac{6\epsilon'}{1-2\epsilon'})f(S) < (6+\epsilon)f(S).$$

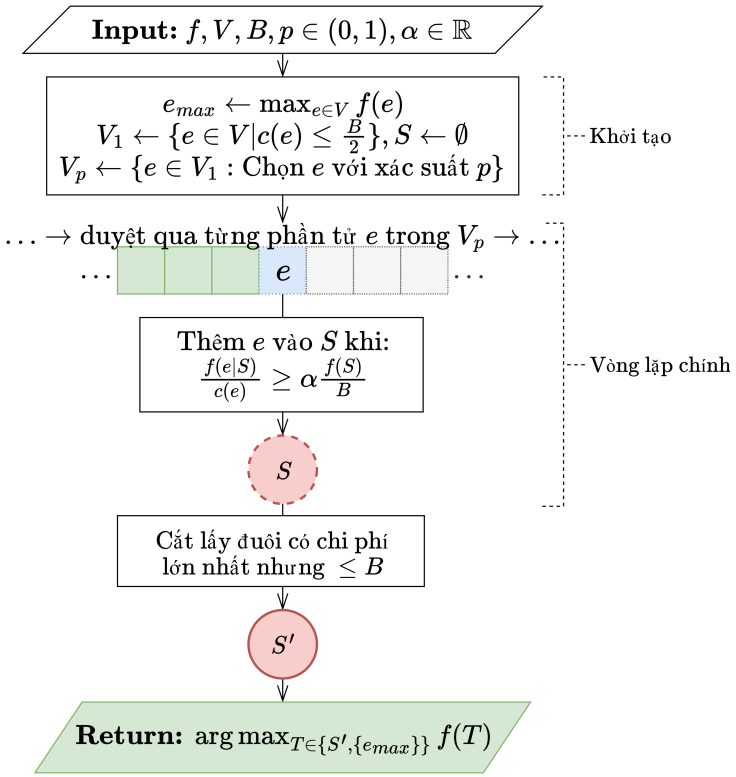
Kết hợp hai trường hợp, ta thu được chứng minh. □

2.3. Thuật toán ngẫu nhiên

Trong hướng tiếp cận thuật toán ngẫu nhiên, luận án đề xuất thuật toán RLA cho bài toán SMK. Thuật toán RLA đạt tỉ lệ xấp xỉ kỳ vọng $\frac{1}{4} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$. So với SMKRANACC [53], RLA giữ cùng hệ số xấp xỉ trong nhóm thuật toán ngẫu nhiên gần tuyến tính nhưng loại bỏ sự phụ thuộc logarit vào kích thước lời giải k trong độ phức tạp truy vấn. RLA được xây dựng từ thuật toán nền tảng LAR. Điểm ngẫu nhiên trong LAR nằm ở bước lấy mẫu tập V_p, còn điểm ngẫu nhiên trong RLA nằm ở bước quyết định có đưa phần tử đủ điều kiện vào dãy nghiệm S_j hay không với xác suất $\frac{1}{2}$. Sau đây, luận án trình bày chi tiết LAR, RLA và phân tích lý thuyết liên quan.

2.3.1. Thuật toán LAR

Thuật toán LAR là phiên bản ngẫu nhiên hóa của Thuật toán 1. Trước hết, thuật toán xác định tập V_1 gồm các phần tử có chi phí không vượt quá $\frac{B}{2}$. Sau đó, mỗi phần tử $e \in V_1$ được chọn độc lập vào tập V_p với xác suất p. Đây là nguồn ngẫu nhiên chính của thuật toán: mỗi lần chạy có thể tạo ra một tập V_p khác nhau, từ đó dẫn tới dãy phần tử được xét và nghiệm cuối cùng khác nhau. Trên tập đã lấy mẫu V_p, thuật toán xây dựng một tập S bằng quy tắc tham lam theo mật độ lợi ích biên, rồi cắt thành đoạn cuối khả thi S' tương tự như cách LA tạo X' và Y'. Cụ thể, các bước được trình bày trong Thuật toán 3 và được minh họa trong Hình 2.3.



Hình 2.3: Lưu đồ hoạt động của thuật toán LAR

Mặc dù LAR là một thuật toán ngẫu nhiên, nhưng nó cung cấp tỉ lệ xấp xỉ tốt hơn so với LA và có thể được sử dụng làm nền tảng cho việc thiết kế thuật toán ngẫu nhiên hóa sau này là RLA.

Tiếp theo, luận án tiến hành phân tích lý thuyết cho thuật toán LAR. **Phác thảo chứng minh.** Ý tưởng chứng minh của LAR là liên hệ nghiệm tối ưu trên tập phần tử nhỏ V_1 với tập ngẫu nhiên V_p được lấy mẫu. Hai bổ đề về tập con ngẫu nhiên cho phép chặn kỳ vọng giá trị của phần giao giữa nghiệm tối ưu và V_p, cũng như giá trị khi ghép tập ngẫu nhiên với nghiệm đang xây dựng. Sau đó, quy tắc tham lam theo mật độ lợi ích biên được dùng để chuyển cận này thành cận theo $\mathbb{E}[f(S)]$, còn bước cắt suffix bảo đảm nghiệm khả thi S' vẫn giữ được một phần đủ lớn của S. Tối ưu các tham số p và α cho ra

Algorithm 3 Thuật toán LAR

Input: An instance (f, V, B), parameters $p \in (0,1), \alpha \in \mathbb{R}$
Output: A feasible solution S_o for SMK, i.e., $c(S_o) \leq B$

$e_{max} \leftarrow \arg\max_{e \in V} f(e)$, $V_1 \leftarrow \{e \in V | c(e) \leq \frac{B}{2}\}$
$V_p \leftarrow \{e \in V_1 :$ Select e with probability $p\}$, $S \leftarrow \emptyset$
foreach $e \in V_p$ **do**
  If $\frac{f(e|S)}{c(e)} \geq \alpha \frac{f(S)}{B}$ **then** $S \leftarrow S \cup \{e\}$
$S' \leftarrow \arg\max_{S(j):0 \leq j \leq |S|, c(S(j)) \leq B} c(S(j))$, where $S(j)$ is the suffix consisting of the last j elements inserted into S and $S(0) = \emptyset$.
$S_o \leftarrow \arg\max_{T \in \{S', \{e_{max}\}\}} f(T)$
return S_o.

hệ số xấp xỉ cuối cùng. Để phục vụ cho quá trình này, luận án nhắc lại hai bổ đề quan trọng đã được chứng minh trong các công trình nghiên cứu trước đây.

Bổ đề 2.4 (Bổ đề 2.2 của Feige và cộng sự [43]). *Cho $f : 2^V \mapsto \mathbb{R}_+$ là một hàm submodular. Ký hiệu $A(p)$ là một tập con ngẫu nhiên của A trong đó mỗi phần tử xuất hiện với xác suất nhiều nhất p (không nhất thiết độc lập). Khi đó ta có $\mathbb{E}[f(A(p))] \geq f(\emptyset) + p \cdot f(A)$.*

Bổ đề 2.5 (Bổ đề 2.2 của Buchbinder và cộng sự [15]). *Cho $f : 2^V \mapsto \mathbb{R}_+$ là một hàm submodular. Ký hiệu $A(p)$ là một tập con ngẫu nhiên của A trong đó mỗi phần tử xuất hiện với xác suất nhiều nhất p (không nhất thiết độc lập). Khi đó ta có $\mathbb{E}[f(A(p))] \geq (1-p)f(\emptyset)$.*

Trước khi chứng minh Định lý 2.3, luận án đưa ra Bổ đề sau, giúp thiết lập mối liên hệ giữa S và S'.

Bổ đề 2.6. $f(S') \geq \frac{\alpha}{\alpha+2} f(S)$.

Chứng minh. Nếu $S' = S$, Bổ đề hiển nhiên đúng. Ta xét trường hợp $S' \subset S$. Ký hiệu $S_1 = S \setminus S'$. Gọi $S = \{s_1, s_2, \ldots, s_{|S|}\}, S^i = \{s_1, s_2, \ldots, s_i\}, 1 \leq i \leq |S|, S^0 = \emptyset$. Giả sử $S_1 = S \setminus S' = \{s_1, \ldots, s_l\}$ và $S' = \{s_{l+1}, s_{l+2}, \ldots, s_{|S|}\}$. Theo quy tắc lựa chọn của mỗi $s_i \in S$, ta dễ thấy $f(S^{i+1}) > f(S^i)$. Với lập luận tương tự như trong chứng minh Định lý 2.1, ta có:

$$\begin{aligned} f(S) - f(S_1) &= \sum_{i=1}^{|S'|} f(s_{l+i} | S_1 \cup \{s_{l+1}, \ldots, s_{l+i-1}\}) \\ &\geq \sum_{i=1}^{|S'|} \frac{c(s_{l+i})\alpha}{B} f(S_1 \cup \{s_{l+1}, \ldots, s_{l+i-1}\}) \\ &\geq \sum_{i=1}^{|S'|} \frac{c(s_{l+i})\alpha}{B} f(S_1) = \frac{c(S')}{B} f(S_1) \end{aligned}$$

$$\geq \frac{\alpha f(S_1)}{2}$$

suy ra $f(S_1) \leq \frac{2}{2+\alpha} f(S)$. Nhờ tính chất submodular của f, ta thu được:

$$f(S') \geq f(S) - f(S_1) \geq \frac{\alpha}{\alpha+2} f(S).$$

Điều này hoàn tất chứng minh. □

Định lý 2.3. *Thuật toán 3 yêu cầu $O(n)$ truy vấn và đạt được tỉ lệ xấp xỉ bằng $\frac{1}{16.034}$ với $p = \sqrt{2} - 1$ và $\alpha = \sqrt{2 + 2\sqrt{2}}$.*

Chứng minh. Thuật toán cần n truy vấn để tìm e_{max} và nhiều nhất n truy vấn để xây dựng S, do đó độ phức tạp truy vấn của thuật toán là $O(n)$. Ký hiệu O_1 là nghiệm tối ưu của bài toán (f, V_1, B) và $O_p = O_1 \cap V_p$. Do cách chọn V_p, mỗi phần tử e trong O_1 xuất hiện trong O_p với xác suất p, nên $\mathbb{E}[c(O_p)] = pc(O_1) \leq pB$ và bằng cách áp dụng Bổ đề 2.4, ta có:

$$\mathbb{E}[f(O_p)] \geq f(\emptyset) + pf(O_1) \geq pf(O_1).$$

Vì mỗi phần tử e trong V xuất hiện trong S với xác suất nhiều nhất p, nên khi áp dụng Bổ đề 2.5 cho $g(\cdot) = f(\cdot \cup O_p)$, ta được:

$$\mathbb{E}[f(S \cup O_p)] = \mathbb{E}[\mathbb{E}[f(S \cup O_p)]|O_p] \geq (1-p)\mathbb{E}[f(O_p)] \geq p(1-p)f(O_1).$$

Theo quy tắc lựa chọn trong Thuật toán 3, ta có $\frac{f(e|S)}{c(e)} \leq \alpha \frac{f(S)}{B}$ với mọi $e \in O_p \setminus S$, do đó:

$$p(1-p)f(O_1) - \mathbb{E}[f(S)] \leq \mathbb{E}[f(S \cup O_p) - f(S)] \leq \mathbb{E}[\sum_{e \in O_p \setminus S} \frac{c(e)\alpha}{B} f(S)]$$

$$\leq \alpha \mathbb{E}[\frac{c(O_p)}{B} f(S)] \leq \alpha p \mathbb{E}[f(S)]$$

suy ra $f(O_1) \leq \frac{1+\alpha p}{p(1-p)} \mathbb{E}[f(S)]$. Theo tính chất submodular của f, ta có:

$$f(O) \leq f(O_1) + f(O_2) \leq \frac{1+\alpha p}{p(1-p)} \mathbb{E}[f(S)] + f(e_{max})$$

$$\leq \left(\frac{(1+\alpha p)(\alpha+2)}{p(1-p)\alpha} + 1 \right) \mathbb{E}[f(S_o)] \quad \text{(Áp dụng Bổ đề 2.6).}$$

Cuối cùng, ta có bất đẳng thức sau:

$$f(O) \leq \left(\frac{(1+\alpha p)(\alpha+2)}{p(1-p)\alpha} + 1 \right) \mathbb{E}[f(S_o)],$$

ta đặt

$$C(\alpha, p) := \frac{(1+\alpha p)(\alpha+2)}{p(1-p)\alpha} + 1,$$

khi đó ta có $f(O) \leq C(\alpha, p)\mathbb{E}[f(S_o)]$, nên hệ số xấp xỉ là $1/C(\alpha, p)$. Do đó, ta lựa chọn các tham số $\alpha > 0$ và $0 < p < 1$ sao cho $C(\alpha, p)$ là nhỏ nhất. Xem $C(\alpha, p)$ như một hàm theo α (với p cố định) và áp dụng điều kiện tối ưu bậc một, ta thu được quan hệ $\alpha^2 p = 2$.

Sau đó, chọn $p = \sqrt{2} - 1$ (khi đó $(1-p)/p = \sqrt{2}$, giúp đơn giản biểu thức $C(\alpha, p)$), từ $\alpha^2 p = 2$ suy ra $\alpha^2 = 2/p = 2(\sqrt{2}+1) = 2 + 2\sqrt{2}$, do đó $\alpha = \sqrt{2+2\sqrt{2}}$. Với lựa chọn $\alpha = \sqrt{2+2\sqrt{2}}$ và $p = \sqrt{2} - 1$, ta nhận được hệ số xấp xỉ mong muốn.

□

2.3.2. Thuật toán RLA

Thuật toán RLA (Thuật toán 4) là thuật toán xấp xỉ ngẫu nhiên đạt tỉ lệ $\frac{1}{4} - \epsilon$ theo kỳ vọng với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$. Thuật toán kế thừa ý tưởng dùng cận ban đầu từ LAR và dùng ngưỡng giảm dần như DLA, nhưng thay hai tập rời nhau bằng một dãy nghiệm ngẫu nhiên S_j. Khi một phần tử vượt qua điều kiện mật độ lợi ích biên và còn khả thi về chi phí, thuật toán đưa phần tử đó vào danh sách ứng viên U; sau đó phần tử chỉ được thêm vào nghiệm kế tiếp với xác suất $\frac{1}{2}$. Vì vậy, tính ngẫu nhiên của RLA không nằm ở việc kiểm tra điều kiện ngưỡng, mà nằm ở bước nhận hoặc loại phần tử đủ điều kiện.

Algorithm 4 Thuật toán RLA

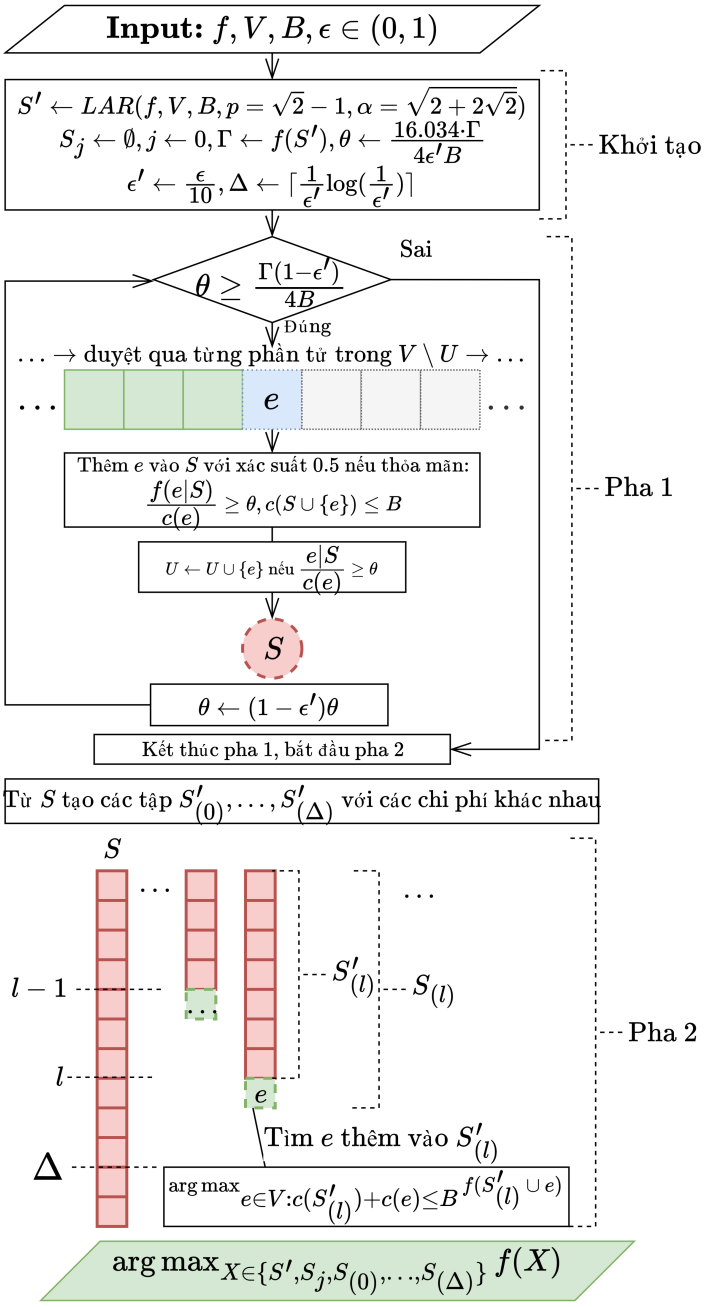
Input: An instance (f, V, B), parameters $\epsilon \in (0, 1)$
Output: A feasible solution S of SMK, i.e., $c(S) \leq B$

$S' \leftarrow \text{LAR}(f, V, B, p = \sqrt{2} - 1, \alpha = \sqrt{2 + 2\sqrt{2}})$
$S_j \leftarrow \emptyset$, $j \leftarrow 0$, $\Gamma \leftarrow f(S')$, $\epsilon' \leftarrow \frac{\epsilon}{10}$, $\theta \leftarrow \frac{16.034 \cdot \Gamma}{4\epsilon' B}$, $\Delta \leftarrow \lceil \frac{1}{\epsilon'} \log(\frac{1}{\epsilon'}) \rceil$
while $\theta \geq \frac{\Gamma(1-\epsilon')}{4B}$ **do**
    foreach $e \in V \setminus \{u_1, u_2, \ldots, u_j\}$ **do**
        if $\frac{f(e|S_j)}{c(e)} \geq \theta$ *and* $c(S_j) + c(e) \leq B$ **then**
            $u_{j+1} \leftarrow e$;
            With probability $\frac{1}{2}$ do: $S_{j+1} \leftarrow S_j \cup \{e\}$ otherwise $S_{j+1} \leftarrow S_j$
            $j \leftarrow j + 1$
    $\theta \leftarrow (1 - \epsilon')\theta$
for $l = 0$ *to* Δ **do**
    $S'_{(l)} \leftarrow \arg\max_{S_i : c(S_i) \leq \epsilon' B(1+\epsilon')^l, 0 \leq i \leq j} i$
    $e^l_{max} \leftarrow \arg\max_{e \in V : c(S'_{(l)} \cup \{e\}) \leq B} f(S'_{(l)} \cup \{e\})$
    $S_{(l)} \leftarrow S'_{(l)} \cup \{e^l_{max}\}$
$S \leftarrow \arg\max_{X \in \{S', S_j, S_{(0)}, \ldots, S_{(\Delta)}\}} f(X)$
return S.

Pha đầu tiên của thuật toán bao gồm một vòng lặp ngưỡng. Mỗi vòng quét qua các phần tử chưa thuộc tập ứng viên $U = \{u_1, \ldots, u_j\}$. Nếu một phần tử e có mật độ lợi ích biên không nhỏ hơn ngưỡng θ và vẫn thỏa ràng buộc chi phí khi ghép với S_j, thuật toán ghi nhận e là phần tử ứng viên tiếp theo u_{j+1}. Sau đó, thuật toán sử dụng một phép chọn ngẫu nhiên: với xác suất $\frac{1}{2}$, đặt $S_{j+1} = S_j \cup \{e\}$; với xác suất còn lại, đặt $S_{j+1} = S_j$. Cơ

chế này giúp mỗi phần tử đủ điều kiện có xác suất xuất hiện được kiểm soát, là cơ sở để áp dụng các bổ đề về tập con ngẫu nhiên trong phân tích.

Trong pha thứ hai, thuật toán tăng cường chất lượng của nghiệm ứng viên S_j bằng cách sử dụng chiến lược tương tự như trong DLA (dòng 10–13). Cụ thể, các bước được trình bày trong Thuật toán 4 và được minh họa trong Hình 2.4.



Hình 2.4: Lưu đồ hoạt động của thuật toán RLA

Tiếp theo, luận án tiến hành phân tích hiệu năng của RLA. Xét tại thời điểm kết thúc của thuật toán, trước tiên ta định nghĩa hoặc nhắc lại một số ký hiệu trong Bảng 2.3.

Bảng 2.3: Bảng các ký tự toán học dùng trong phân tích thuật toán RLA

Ký hiệu	**Ý nghĩa**
O	Lời giải tối ưu của bài toán SMK và $\mathsf{opt} = f(O)$
U	Danh sách các phần tử vượt qua điều kiện ngưỡng trong pha 1, được ký hiệu theo thứ tự xuất hiện là $U = \{u_1, u_2, \ldots, u_j\}$
$\tau(e)$	Chỉ số thời điểm phần tử e được đưa vào danh sách U: nếu $e = u_i \in U$ thì $\tau(e) = i$ và $S^{<e} = S_{i-1}$; nếu $e \notin U$ thì $\tau(e) = +\infty$
T_j	Chỉ số tiền tố dùng trong phân tích với danh sách $U = \{u_1, \ldots, u_j\}$: nếu $c(S_{j-1} \cup \{u_j\}) \leq B - c(r)$ thì $T_j = j$; ngược lại $T_j = \min\{i : 0 \leq i \leq j-1,\ c(S_i \cup \{u_{i+1}\}) > B - c(r)\}$
$X_{e,i}, Y_{e,i}$	Các biến chỉ báo trong Bổ đề 2.7; chỉ số i cho biết các biến này được xét tương ứng với trạng thái S_i

Bảng 2.3 dùng ký hiệu T_j thay cho T để nhấn mạnh rằng chỉ số này phụ thuộc vào danh sách ứng viên sau pha 1. Tương tự, các biến chỉ báo được viết là $X_{e,i}$ và $Y_{e,i}$ vì chúng phụ thuộc đồng thời vào phần tử e và trạng thái S_i đang xét.

Phác thảo chứng minh. Phân tích của RLA kết hợp hai nguồn cận: cận ngẫu nhiên từ việc mỗi phần tử đủ điều kiện chỉ được đưa vào nghiệm với xác suất $\frac{1}{2}$, và cận chi phí từ pha chọn tiền tố tương tự DLA. Bổ đề chính theo dõi trạng thái S_i sau từng phần tử ứng viên, qua đó chặn giá trị của phần nghiệm tối ưu đã được xét và phần chưa được xét. Sau khi có cận cho mọi tiền tố, định lý tách theo trường hợp phần tử lớn nhất r có chiếm gần hết ngân sách hay không, rồi dùng các nghiệm tăng cường ở pha 2 để đạt hệ số $\frac{1}{4} - \epsilon$ theo kỳ vọng.

Bổ đề 2.7 cung cấp một công cụ hữu ích để ước lượng $f(S_i)$ với mọi $i \leq j$, từ đó hỗ trợ phân tích và chứng minh hiệu năng của RLA.

Bổ đề 2.7. *Với mỗi $u_i \in \{u_1, \ldots, u_j\}$, ta định nghĩa: $O_{\leq i} = \{e : e \in O, \tau(e) \leq i\}$, $O_{>i} = \{e : e \in O, \tau(e) > i\}$ và*

$$X_{e,i} = \begin{cases} 1, & e \in O_{\leq i} \setminus S_i \textit{ hoặc } e \in S_i \setminus O, \\ 0, & \textit{ngược lại}. \end{cases}$$

$$Y_{e,i} = \begin{cases} 1, & e \in O_{\leq i} \setminus (S_i \cup \{r\}) \textit{ hoặc } u \in S_i \setminus (O \setminus \{r\}), \\ 0, & \textit{ngược lại}. \end{cases}$$

a) Với mọi S_i ta có $\mathbb{E}[f(S_i)] = \mathbb{E}[\sum_{e \in V} X_{e,i} \cdot f(e|S^{<e})]$.

b) Với mọi S_i thỏa mãn $c(S_i) \leq B - c(r)$ ta có $\mathbb{E}[f(S_i)] = \mathbb{E}[\sum_{e \in V} Y_{e,i} \cdot f(e|S^{<e})]$.

Chứng minh. Chứng minh này được dựa trên Bổ đề 3 của Han và cộng sự [53].

Chứng minh a. Với mọi $e \in V$, ta định nghĩa:

$$R_e = \begin{cases} f(e|S^{<e}), & e \in S_i, \\ 0, & \text{ngược lại.} \end{cases}$$

Theo định nghĩa của R_e, ta có $f(S_i) = \sum_{e \in V} R_e$. Ta sẽ chỉ ra $\mathbb{E}[R_e] = \mathbb{E}[X_{e,i} \cdot f(e|S^{<e})]$. Với mọi $e \in V$, ta định nghĩa một biến cố bất kỳ để mô tả quá trình ngẫu nhiên của thuật toán cho đến thời điểm e được xét tại Dòng 5 của Thuật toán 4. Ta xét các khả năng đối với e thông qua quá trình ngẫu nhiên $\mathcal{E}_e$.

Trường hợp 1. $\mathcal{E}_e$ là biến cố e không vượt qua điều kiện tại Dòng 5 của Thuật toán 4. Khi đó: $\mathbb{E}[R_e|\mathcal{E}_e] = \mathbb{E}[X_{e,i} \cdot f(e|S^{<e})|\mathcal{E}_e] = 0$.

Trường hợp 2. $\mathcal{E}_e$ là biến cố e vượt qua điều kiện tại Dòng 5 của Thuật toán 4. Với $\mathcal{E}_e$ thì $S^{<e}$ là cố định và e là ngẫu nhiên. Vì e được chọn vào S_i với xác suất $\frac{1}{2}$, ta có $\mathbb{E}[R_e|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$. Ta sẽ chỉ ra: $\mathbb{E}[X_{e,i} \cdot f(e|S^{<e})|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$ bằng cách xét hai trường hợp sau:

- Nếu $e \in O \Rightarrow e \notin S_i \setminus O$. Nếu e được thêm vào S_i (tức là không bị loại bỏ), khi đó $e \in S_i \cap O$ và $X_{e,i} = 0$. Nếu e không được thêm vào S_i, khi đó $e \in O_{\leq i} \setminus S_i$ và do đó $X_{e,i} = 1$. Suy ra $\mathbb{E}[X_{e,i} \cdot f(e|S^{<e})|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$.

- Nếu $e \notin O$, ta có $e \notin O_{\leq i} \setminus S_i$. Nếu e được chọn vào S_i (tức là không bị loại bỏ), khi đó $e \in S_i \setminus O$ và $X_{e,i} = 1$. Nếu e bị loại bỏ, khi đó $e \notin O \cup S_i$, do đó $X_{e,i} = 0$. Suy ra $\mathbb{E}[X_{e,i} \cdot f(e|S^{<e})] = \frac{1}{2} f(e|S^{<e})$.

Kết hợp hai trường hợp trên, ta có điều phải chứng minh.

Chứng minh b. Tương tự như phần trước, với mọi $e \in V$, ta định nghĩa

$$R_e = \begin{cases} f(e|S^{<e}), & e \in S_i, \\ 0, & \text{ngược lại.} \end{cases}$$

Ta có $f(S_i) = \sum_{e \in V} f(e|S^{<e})$. Ta sẽ chỉ ra: $\mathbb{E}[R_e] = \frac{1}{2}\mathbb{E}[Y_{e,i} \cdot f(e|S^{<e})]$.

Ta xét các khả năng đối với e thông qua một quá trình ngẫu nhiên $\mathcal{E}_e$:

Trường hợp 1. $\mathcal{E}_e$ là biến cố e không bao giờ được xét tại Dòng 5 của Thuật toán 4 hoặc $c(S^{<e}) + c(e) > B - c(r)$. Khi đó: $\mathbb{E}[R_e|\mathcal{E}_e] = \mathbb{E}[Y_{e,i} \cdot f(e|S^{<e})|\mathcal{E}_e] = 0$.

Trường hợp 2. $\mathcal{E}_e$ là biến cố e được xét bởi thuật toán và $c(S^{<e}) + c(e) \leq B - c(r)$. Khi đó ta cũng có $\mathbb{E}[R_e|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$ bằng cách xét hai tình huống:

- Nếu $e \in O \setminus \{r\}$, khi đó $e \notin S_i \setminus (O \setminus \{r\})$. Nếu e không bị loại bỏ, khi đó $e \in S_i \cap (O \setminus \{r\})$ và $Y_{e,i} = 0$. Nếu e bị loại bỏ, khi đó $e \in O_{\leq i} \setminus (S_i \cup \{r\})$. Do đó $Y_{e,i} = 1$. Vì vậy, $\mathbb{E}[Y_{e,i} f(e|S^{<e})|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$.

- Nếu $e \notin O \setminus \{r\}$, khi đó $e \notin O_{\leq i} \setminus (S_i \cup \{r\})$. Nếu e không bị loại bỏ, khi đó $e \in S_i \setminus (O \setminus \{r\})$ và $Y_{e,i} = 1$. Nếu e bị loại bỏ thì $Y_{e,i} = 0$. Do đó, $\mathbb{E}[Y_{e,i} f(e|S^{<e})|\mathcal{E}_e] = \frac{1}{2} f(e|S^{<e})$.

Điều này hoàn tất chứng minh. □

Định lý 2.4. *Với mọi $\epsilon \in (0,1)$, RLA là một thuật toán ngẫu nhiên có độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$ và đạt được tỉ lệ xấp xỉ $\frac{1}{4}-\epsilon$ theo kỳ vọng.*

Chứng minh. Độ phức tạp truy vấn của RLA được xác định bằng lập luận tương tự như trong chứng minh Định lý 2.2. Ký hiệu θ_i là θ tại vòng lặp thứ i, $\theta_{(i)}$ là θ khi u_i được thêm vào U và θ_{last} là θ tại vòng lặp cuối cùng của vòng lặp thứ nhất.

Để chứng minh hệ số xấp xỉ, ta xét các trường hợp sau:

Trường hợp 1. Nếu $c(r) > (1-\epsilon')B$, khi đó $c(O\setminus\{r\}) < B-(1-\epsilon')B = \epsilon' B$. Ta xét hai trường hợp con:

Trường hợp 1.1. Nếu $c(S_j) \geq (1-\epsilon')B$, khi đó $f(S) \geq f(S_j) \geq c(S_j)(1-\epsilon')\frac{\mathsf{opt}}{4} \geq (1-\epsilon')^2\frac{\mathsf{opt}}{4}$.

Vì $\epsilon' = \frac{\epsilon}{10} < \frac{1}{10}$, ta có: $\mathsf{opt} \leq \frac{4f(S)}{(1-\epsilon')^2} \leq 4(1+\frac{10}{9}\epsilon')^2 f(S) \leq (4+\epsilon)f(S)$.

Trường hợp 1.2. Nếu $c(S_j) < (1-\epsilon')B$, khi đó $c(S_j)+c(e) \leq c(S_j)+c(O\setminus\{r\}) < B$ với mọi $e \in (O\setminus\{r\})\setminus S_j$. Do đó $\frac{f(e|S_j)}{c(e)} < \frac{(1-\epsilon')\Gamma}{4B} \leq \frac{(1-\epsilon')\mathsf{opt}}{4B}$. Suy ra:

$$\begin{aligned} f((O\setminus\{r\})\cup S_j) - f(S_j) &\leq \sum_{e\in(O\setminus\{r\})\setminus S_j} f(e|S_j) \\ &< c(O\setminus\{r\})(1-\epsilon')\frac{\mathsf{opt}}{4B} \leq \epsilon'(1-\epsilon')\frac{\mathsf{opt}}{4}. \end{aligned} \tag{2.15}$$

Vì mỗi phần tử trong V xuất hiện trong S_j với xác suất $\frac{1}{2}$, áp dụng Bổ đề 2.5 ta có $\mathbb{E}[f(O\setminus\{r\}\cup S_j)] \geq \frac{1}{2}f(O\setminus\{r\})$. Kết hợp với (2.15), ta được: $f(O) \leq f(O\setminus\{r\})+f(r) \leq 2\mathbb{E}[f(O\setminus\{r\}\cup S_j)] + f(e_{max}) < 3\mathbb{E}[f(S)] + \frac{\epsilon'(1-\epsilon')\mathsf{opt}}{2}$. Suy ra:

$$\mathsf{opt} < \frac{6\mathbb{E}[f(S)]}{2-\epsilon'(1-\epsilon')} \leq \frac{6\mathbb{E}[f(S)]}{2-\epsilon'} \leq (4+\epsilon)\mathbb{E}[f(S)].$$

Trường hợp 2. Nếu $c(r) \leq (1-\epsilon')B$, khi đó $c(O\setminus\{r\}) \geq \epsilon' B$. Ta xét các trường hợp con sau:

Trường hợp 2.1. Nếu $T_j = j$, theo định nghĩa của T_j ta có $O_{>T_j} = \emptyset$. Do đó

$$\begin{aligned} f(S_{T_j}\cup O) - f(S_{T_j}) &\leq \sum_{e\in O_{\leq T_j}\setminus S_{T_j}} f(e|S_{T_j}) \\ &\leq \sum_{e\in O_{\leq T_j}\setminus S_{T_j}} f(e|S^{<e}) + \sum_{e\in S_{T_j}\setminus O} f(e|S^{<e}) \qquad (2.16) \\ &= \sum_{e\in V} X_{e,T_j}\cdot f(e|S^{<e}) = \mathbb{E}[f(S_{T_j})]. \qquad (2.17) \end{aligned}$$

Trong đó (2.16) là do $f(e|S^{<e}) > 0$, với mọi $e \in S_j$ và X_{e,T_j} được định nghĩa trong Bổ đề 2.7. Áp dụng lại Bổ đề 2.5, ta có $\mathbb{E}[f(O\cup S_{T_j})] \geq \frac{f(O)}{2}$. Kết hợp với (2.17), ta thu được: $\mathbb{E}[f(S)] \geq \mathbb{E}[f(S_{T_j})] \geq \frac{\mathbb{E}[f(O\cup S_{T_j})]}{2} \geq \frac{f(O)}{4}$.

Trường hợp 2.2. Nếu $T_j < j$, khi đó U chứa ít nhất $T_j + 1$ phần tử và ta có $c(S_{T_j}) + c(u_{T_j+1}) > B - c(r) > \epsilon' B$.

Bây giờ ta xét vòng lặp thứ hai của Thuật toán 2. Vì $\epsilon' B < B - c(r) \leq B$, tồn tại một số nguyên l sao cho

$$\epsilon' B \leq (1+\epsilon')^l \epsilon' B \leq B - c(r) < (1+\epsilon')^{l+1} \epsilon' B.$$

Giả sử $S'_{(l)} = S_i$ với một i. Theo quy tắc lựa chọn $S'_{(l)}$, ta có $c(S_i) \leq (1+\epsilon')^l \epsilon' B < c(S_i \cup \{u_{i+1}\})$, do đó $c(S_i \cup \{u_{i+1}\}) > \frac{\epsilon' B}{1+\epsilon'}$.

Ta xét hai trường hợp con:

Nếu u_{i+1} được xét tại vòng lặp đầu tiên, theo quy tắc chọn e^l_{max} ở vòng lặp thứ hai, ta có:

$$f(S_{(l)}) \geq f(S_i \cup \{u_{i+1}\}) \geq c(S_i \cup \{u_{i+1}\})\theta_1 \geq \frac{\mathsf{opt}}{4(1+\epsilon')^2}.$$

Suy ra, $\mathsf{opt} \leq 4(1+\epsilon')^2 f(S) < (4+\epsilon) f(S)$.

Nếu u_{i+1} được xét tại vòng lặp thứ l, $l \geq 2$. Đặt $\hat{S} = S_i \setminus (O \setminus \{r\})$ và $\hat{O} = O_{\leq i} \setminus (S_i \cup \{r\})$. Ta chứng minh:

$$c(\hat{S}) + c(u_{i+1}) > c(O_{>i} \setminus \{r\}). \tag{2.18}$$

Có thể thấy:

$$\begin{aligned} c(S_i \setminus (O \setminus \{r\})) + c(S_i \cap (O \setminus \{r\})) + c(u_{i+1}) &= c(S_i) + c(u_{i+1}) > B - c(r) \geq c(O \setminus \{r\}) \\ &\geq c(O_{>i} \setminus \{r\}) + c(S_i \cap (O \setminus \{r\})), \end{aligned}$$

nên (2.18) đúng.

Mặt khác, với mọi $e \in O_{>i} \setminus \{r\}$, độ tăng biên của nó so với S_i nhỏ hơn ngưỡng tại vòng lặp trước, tức là $\frac{f(e|S_i)}{c(e)} \leq \frac{\theta_{(i+1)}}{1-\epsilon'}$. Kết hợp với (2.18), ta thu được:

$$\begin{aligned} \sum_{e \in O_{>i} \setminus \{r\}} f(e|S_i) &= \sum_{e \in O_{>i} \setminus \{r\}} \frac{f(e|S_i)}{c(e)} c(e) \\ &\leq \frac{c(O_{>i} \setminus \{r\})\theta_{(i+1)}}{1-\epsilon'} < \frac{c(\hat{S} \cup \{u_{i+1}\})\theta_{(i+1)}}{1-\epsilon'} \\ &\leq \frac{\sum_{e \in \hat{S} \cup \{u_{i+1}\}} f(e|S^{<e})}{1-\epsilon'}, \end{aligned} \tag{2.19}$$

trong đó (2.19) do $\frac{f(e|S^{<e})}{c(e)} \geq \theta_{(i+1)}$, với mọi $e \in S_i \cup \{u_{i+1}\}$. Suy ra:

$$\begin{aligned} f(S_i \cup O) - f(S_i \cup \{r\}) &\leq \sum_{e \in O \setminus (S_i \cup \{r\})} f(e|S_i) \\ &= \sum_{e \in \hat{O}} f(e|S_i) + \sum_{e \in O_{>i} \setminus \{r\}} f(e|S_i) \qquad (2.20) \\ &< \frac{\sum_{e \in \hat{O}} f(e|S^{<e}) + \sum_{e \in \hat{S}} f(e|S^{<e}) + f(u_{i+1}|S_i)}{1-\epsilon'} \qquad (2.21) \end{aligned}$$

$$\leq \frac{Y_{e,i} \cdot f(e|S^{<e}) + f(e^l_{max}|S_i)}{1-\epsilon'} \tag{2.22}$$

trong đó $Y_{e,i}$ được định nghĩa trong Bổ đề 2.7.

Từ (2.22) và áp dụng Bổ đề 2.7, ta có:

$$\mathbb{E}[f(S_i \cup O)] < \frac{\mathbb{E}[f(S_i)] + \mathbb{E}[f(e^l_{max}|S_i)]}{1-\epsilon'} +$$
$$\mathbb{E}[f(S_i \cup \{r\})] \leq \frac{\mathbb{E}[f(S)]}{1-\epsilon'} + \mathbb{E}[f(S)] = \frac{2-\epsilon'}{1-\epsilon'}\mathbb{E}[f(S)].$$

Áp dụng Bổ đề 2.5, ta có $f(O) \leq 2\mathbb{E}[f(S_i \cup O)]$. Do đó

$$f(O) < \frac{2(2-\epsilon')}{1-\epsilon'}\mathbb{E}[f(S)] \leq (4+\epsilon)\mathbb{E}[f(S)].$$

Kết hợp tất cả các trường hợp, ta hoàn tất chứng minh. □

2.4. Thuật toán song song

Trong hướng tiếp cận thuật toán song song, luận án đề xuất thuật toán AST cho bài toán SMK. Thuật toán AST đạt hệ số xấp xỉ kỳ vọng $\frac{1}{7} - \epsilon$, duy trì độ phức tạp song song $O(\log n)$ và độ phức tạp truy vấn $\tilde{O}(nk)$. So với thuật toán ParSKP1 của Cui và cộng sự [30], AST cải thiện hệ số xấp xỉ từ $\frac{1}{8} - \epsilon$ lên $\frac{1}{7} - \epsilon$ trong cùng mức độ phức tạp song song $O(\log n)$. So với thuật toán ParSKP2 có hệ số tốt hơn $\frac{1}{7.83} - \epsilon$ nhưng cần $O(\log^2 n)$ vòng song song, AST ưu tiên giữ số vòng song song ở mức $O(\log n)$. Ý tưởng chính của AST là xây dựng hai tập ứng viên X và Y theo cơ chế ngưỡng luân phiên: ở các vòng lẻ cập nhật X, ở các vòng chẵn cập nhật Y, sau đó tạo thêm các nghiệm khả thi từ tiền tố của hai tập này.

2.4.1. Một số khái niệm liên quan

Để xây dựng AST, luận án sử dụng hai bài toán con. Bài toán thứ nhất là tối đa hàm submodular không ràng buộc, được dùng trong pha cuối để xử lý phần tử có chi phí rất nhỏ trong V_0. Bài toán thứ hai là ngưỡng mật độ lợi ích biên, được dùng trong thủ tục RandBatch để lọc song song các phần tử có mật độ đủ lớn khi cập nhật X hoặc Y. Hai bài toán con được phát biểu như sau:

Định nghĩa 2.2 (Bài toán tối đa hàm submodular không ràng buộc - UnSubMax [43])**.** Bài toán yêu cầu tìm tập con $S \subseteq V$ sao cho $f(S)$ là lớn nhất, không có bất kỳ ràng buộc nào.

Bài toán này đã được chứng minh là NP-khó [43]. Để đạt được hệ số xấp xỉ đã nêu, luận án sử dụng thuật toán có độ phức tạp song song thấp của Chen và cộng sự [21], đạt hệ số xấp xỉ $(\frac{1}{2} - \epsilon)$ trong số vòng song song hằng số $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ và số lượng truy vấn tuyến tính $O(\frac{n}{\epsilon^4}\log^3(\frac{1}{\epsilon}))$.

Định nghĩa 2.3 (Ngưỡng mật độ lợi ích biên - DS)**.** Bài toán nhận một bộ (f, V, B), một ngưỡng cố định τ và tham số $\epsilon > 0$, yêu cầu tìm tập con $S \subseteq V$ thỏa mãn hai điều kiện: (1) $f(S) \geq c(S) \cdot \tau$; (2) $\sum_{e \in V \setminus S} f(e|S) \leq \epsilon \cdot \mathsf{opt}$.

Hai thuật toán trong tài liệu [2] và [30] đảm bảo các điều kiện trên. Trong luận án này, nghiên cứu sinh sử dụng lại thuật toán RandBatch(Thuật toán 5) được giới thiệu bởi Cui và cộng sự [30] để giải cho bài toán ngưỡng mật độ lợi ích biên. Thuật toán RandBatch nhận vào tập I, hàm submodular $f(\cdot)$ và các tham số ϵ, M để điều chỉnh độ chính xác và độ phức tạp của lời giải. RandBatch được kết hợp với ngưỡng mật độ lợi ích biên để thiết lập cơ chế lọc song song cho bài toán SMK. Hiệu năng của RandBatch được thể hiện trong các Bổ đề sau.

Algorithm 5 Thuật toán RandBatch [30]

Input: θ, I, M, $p \in (0, 1]$, $\epsilon \in (0, 1)$, $f(\cdot)$, $c(\cdot)$

$A \leftarrow \emptyset, U \leftarrow \emptyset, count \leftarrow 0$
$L \leftarrow \{u \in I : \frac{f(u|A)}{c(u)} \geq \theta \wedge c(A \cup \{u\}) \leq B\}$;
while $L \neq \emptyset \wedge count < M$ **do**
  $\{v_1, v_2, \ldots, v_d\} \leftarrow GetSEQ(A, L, c(\cdot))$;
  foreach $i \in \{0, 1, \ldots, d\}$ **do**
    $V_i \leftarrow \{v_1, v_2, \ldots, v_i\}, G_i \leftarrow A \cup V_i$;
    $E_i^+ \leftarrow \{u \in L : \frac{f(u|G_i)}{c(u)} \geq \theta \wedge c(G_i \cup \{u\}) \leq B\}$;
    $E_i^- \leftarrow \{u \in L : f(u|G_i) < 0\}$;
    $D_i \leftarrow \{v_j : j \in [i] \wedge f(v_j|A \cup V_{j-1}) < 0\}$;
  Find $t_1 \leftarrow \min_{i \leq d}\{c(E_i^+) \leq (1 - \epsilon)c(L)\}$, $t_2 \leftarrow \min_{i \leq d}\{\epsilon \sum_{u \in E_i^+} f(u|G_i) \leq \sum_{u \in E_i^-} |f(u|G_i)| + \sum_{v_j \in D_i} |f(v_j|A \cup V_{j-1})|\}$
  $t^* \leftarrow \min\{t_1, t_2\}$; $U \leftarrow U \cup V_{t^*}$;
  With probability p do:
    $A \leftarrow A \cup V_{t^*}$
    If $t_2 \leq t_1$ **then** $count \leftarrow count + 1$
  $L \leftarrow \{u \in L \setminus U : \frac{f(u|A)}{c(u)} \geq \theta \wedge c(A \cup \{u\}) \leq B\}$
return (A, U, L)

Algorithm 6 Thủ tục $GetSEQ(A, I, c(\cdot))$[30]

Input: A base set A, a candidate set I, and cost function $c(\cdot)$

$X \leftarrow I$, $Q \leftarrow \emptyset$, $i \leftarrow 1$
while $X \neq \emptyset$ **do**
  Draw a_i uniformly at random from X
  $Q \leftarrow [a_1, a_2, \ldots, a_i]$
  $X \leftarrow \{e \in X \setminus \{a_i\} : c(e) + c(Q) + c(A) \leq B\}$
  $i \leftarrow i + 1$
return$Q$

Bổ đề 2.8 (Bổ đề 1 của Cui và cộng sự [30]). *Các tập A, L được sinh bởi RandBatch$(\theta, I, M, p, \epsilon, f(\cdot), c(\cdot))$ thỏa mãn $\mathbb{E}[f(A)] \geq (1-\epsilon)^2\theta \cdot \mathbb{E}[c(A)]$ và $\epsilon \cdot M \cdot \sum_{u \in L} f(u|A) \leq \mathsf{opt}$.*

Bổ đề 2.9 (Bổ đề 2 Cui và cộng sự [30]). *RandBatch có $O(\frac{1}{\epsilon p}(\log(|I| \cdot \beta(I)) + M))$ độ phức tạp song song và độ phức tạp truy vấn của nó là $O(|I|k)$ lần độ phức tạp song song, trong đó $\beta(I) = \max_{u,v} \frac{c(u)}{c(v)}$. Nếu ta sử dụng tìm kiếm nhị phân tại Dòng 10 của RandBatch, thì nó có $O(\frac{1}{\epsilon p}(\log(|I| \cdot \beta(I)) + M)\log(k))$ độ phức tạp song song và độ phức tạp truy vấn của nó là $O(|I|)$ lần độ phức tạp song song.*

Có thể sử dụng tìm kiếm nhị phân tại bước xác định t_1, t_2 vì các điều kiện dừng trong thủ tục RandBatch được kiểm tra trên dãy tiền tố $V_i = \{v_1, \ldots, v_i\}$ sinh bởi $GetSEQ$. Khi i tăng, tập tiền tố lớn dần và các đại lượng tương ứng có thể được kiểm tra theo thứ tự tiền tố, nên thuật toán không cần duyệt tuyến tính toàn bộ các giá trị i trong mỗi lần gọi thủ tục. Cách này làm tăng số vòng song song bởi một thừa số logarit nhưng giảm số truy vấn cần dùng trong mỗi vòng. Bên cạnh đó, luận án còn khai thác thêm một tính chất hữu ích của RandBatch khi áp dụng trong thuật toán AST.

Bổ đề 2.10. *Các tập A, L được sinh bởi RandBatch$(\theta, I, M, p, \epsilon, f(\cdot), c(\cdot))$ thỏa mãn $\mathbb{E}[f(a_i|A_{i-1})] \geq (1-\epsilon)^2\mathbb{E}[c(a_i)]\theta$, với $A = \{a_1, a_2, \ldots, a_{|A|}\}, A_i = \{a_1, a_2, \ldots, a_i\}$.*

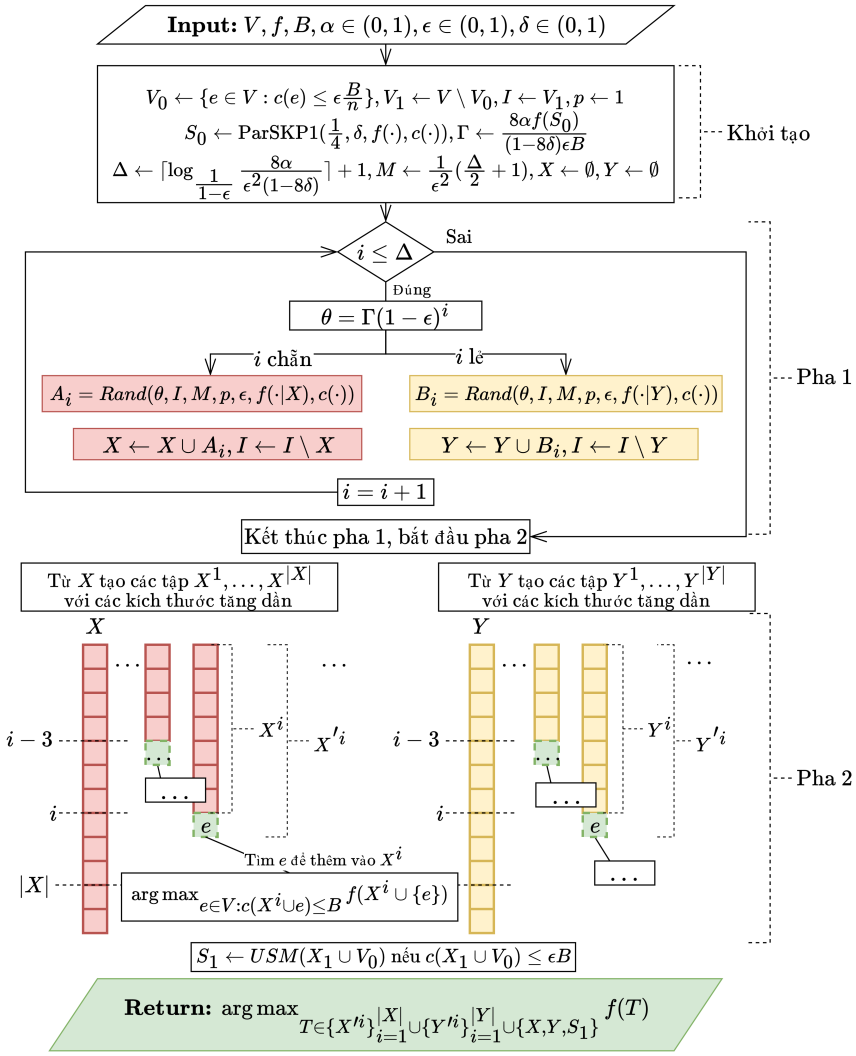
2.4.2. Thuật toán AST

Thuật toán AST (Thuật toán 7) nhận đầu vào là một bộ ba (f, V, B) cùng các tham số δ, ϵ, α và có hai pha chính. Điểm cần lưu ý là việc chia V thành V_0 và V_1 không phải là một bước phụ độc lập: V_1 là tập phần tử được xử lý trong vòng lặp ngưỡng luân phiên, còn V_0 được giữ lại để xử lý bằng thủ tục UnSubMax ở pha thứ hai khi phần chi phí của $X_1 \cup V_0$ còn nhỏ.

Trong pha đầu tiên, thuật toán chia tập cơ sở V thành $V_0 = \{e \in V : c(e) \leq \epsilon B/n\}$ và $V_1 = V \setminus V_0$. Thuật toán chỉ đưa V_1 vào tập làm việc I của vòng lặp chính. Sau đó, thuật toán gọi thủ tục ParSKP1 từ công trình của Cui và cộng sự [30] để thu được nghiệm ban đầu S_0, qua đó thiết lập giá trị Γ dùng cho dãy ngưỡng. Trong vòng lặp chính, AST cập nhật luân phiên hai tập ứng viên: nếu i lẻ thì đặt ngưỡng θ^X và gọi RandBatch trên hàm biên $f(\cdot|X)$ để sinh A_i, sau đó cập nhật $X \leftarrow X \cup A_i$; nếu i chẵn thì làm tương tự với Y thông qua B_i. Sau mỗi lần cập nhật, các phần tử đã được đưa vào tập hiện tại được loại khỏi I, nhờ đó hai tập X và Y được xây dựng theo hai lượt xen kẽ.

Pha thứ hai tạo các nghiệm khả thi từ các tập đã xây dựng. Nếu $c(X_1 \cup V_0) \leq \epsilon B$, thuật toán gọi UnSubMax trên $X_1 \cup V_0$ để khai thác nhóm phần tử có chi phí rất nhỏ. Tiếp theo, thuật toán xét các tiền tố X^i và Y^i theo thứ tự phần tử được thêm vào, rồi ghép mỗi tiền tố với một phần tử khả thi tốt nhất. Nghiệm cuối cùng là tập có giá trị hàm mục tiêu lớn nhất trong các ứng viên này cùng với X, Y, S_1. Cụ thể, các bước được trình bày trong Thuật toán 7 và được minh họa trong Hình 2.5.

AST được xây dựng theo khung thuật toán tham lam ngưỡng luân phiên, nơi hai lời giải được cập nhật luân phiên qua các vòng lặp. Khung này khác biệt với các thuật toán gần đây [2, 30], vốn cập nhật cả hai tập lời giải trong một vòng song song duy nhất, và cũng khác với phương pháp *"twin greedy"* trong [52], nơi cả hai tập được cập nhật đồng thời. Phân tích lý thuyết của AST tập trung vào (1) mối liên hệ giữa X và Y sau mỗi vòng và (2) vai trò của phần tử chi phí lớn nhất r nhằm kiểm soát hệ số xấp xỉ.



Hình 2.5: Lưu đồ hoạt động của thuật toán AST

Xem xét hai tập X và Y sau khi kết thúc pha 1, tức là sau vòng lặp ở các dòng 4–12. Vì thuật toán cập nhật X ở vòng lẻ và cập nhật Y ở vòng chẵn, các ký hiệu dùng trong phân tích cần phân biệt trạng thái của tập sau từng vòng lặp với tiền tố theo thứ tự phần tử được thêm vào. Trước tiên, nghiên cứu sinh định nghĩa hoặc nhắc lại một số ký hiệu được sử dụng trong phân tích thuật toán trong Bảng 2.4.

Phân tích của luận án tập trung vào việc khai thác mối quan hệ giữa hai tập ứng viên X và Y trong quá trình xây dựng luân phiên. Đặc biệt, sau mỗi vòng lặp, chỉ một trong hai tập được cập nhật, do đó giá trị cận dưới cho mỗi vòng có thể được xác lập

Algorithm 7 Thuật toán AST

Input: An instance (f, V, B), parameters α, ϵ, $\delta \in (0, 1)$
Output: A feasible solution S of SMK, i.e., $c(S) \leq B$

$V_0 \leftarrow \{e \in V : c(e) \leq \epsilon\frac{B}{n}\}$, $V_1 \leftarrow V \setminus V_0$, $I \leftarrow V_1$, $p \leftarrow 1$
$S_0 \leftarrow$ ParSKP1$(\frac{1}{4}, \delta, f(\cdot), c(\cdot))$, $\Gamma \leftarrow \frac{8\alpha f(S_0)}{(1-8\delta)\epsilon B}$
$X \leftarrow \emptyset, Y \leftarrow \emptyset$, $\Delta \leftarrow \lceil \log_{\frac{1}{1-\epsilon}} \frac{8\alpha}{\epsilon^2(1-8\delta)} \rceil + 1$, $M \leftarrow \frac{1}{\epsilon^2}(\frac{\Delta}{2} + 1)$
for $i = 1$ *to* Δ **do**
    if i *is* ***odd*** **then**
        $\theta^X \leftarrow \Gamma(1-\epsilon)^i$
        $(A_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^X, I, M, p, \epsilon, f(\cdot|X), c(\cdot))$
        $X \leftarrow X \cup A_i$, $I \leftarrow I \setminus X$
    else
        $\theta^Y \leftarrow \Gamma(1-\epsilon)^i$
        $(B_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^Y, I, M, p, \epsilon, f(\cdot|Y), c(\cdot))$
        $Y \leftarrow Y \cup B_i$, $I \leftarrow I \setminus Y$
For $T \in \{X, Y\}$, define T_i as the state of T after iteration i of the first loop, and T^i as the prefix consisting of the first i elements inserted into T.
if $c(X_1 \cup V_0) \leq \epsilon B$ **then**
    $S_1 \leftarrow$ UnSubMax$(X_1 \cup V_0)$
for $i = 1$ *to* $|X|$ **do**
    $a_i \leftarrow \arg\max_{e \in V : c(X^i \cup e) \leq B} f(X^i \cup \{e\})$, $X'^i \leftarrow X^i \cup \{a_i\}$
for $i = 1$ *to* $|Y|$ **do**
    $b_i \leftarrow \arg\max_{e \in V : c(Y^i \cup e) \leq B} f(Y^i \cup \{e\})$, $Y'^i \leftarrow Y^i \cup \{b_i\}$
$S \leftarrow \arg\max_{T \in \{X'^i\}_{i=1}^{|X|} \cup \{Y'^i\}_{i=1}^{|Y|} \cup \{X, Y, S_1\}} f(T)$
return S

riêng biệt và cộng gộp sau cùng. Một yếu tố then chốt là nhận định khi loại bỏ phần tử có chi phí lớn nhất r, ta có thể kiểm soát chặt chẽ hơn tỉ lệ xấp xỉ tổng thể, nhờ vào việc giới hạn đóng góp biên thấp của phần tử này trong nghiệm tối ưu. Các bổ đề tiếp theo sẽ trình bày chi tiết từng bước đánh giá giá trị hàm mục tiêu của các tập ứng viên so với nghiệm tối ưu đã điều chỉnh, từ đó dẫn đến kết quả cuối cùng về hệ số xấp xỉ của thuật toán.

Bảng 2.4: Bảng các ký tự toán học dùng trong phân tích thuật toán AST

Ký hiệu	**Ý nghĩa**
X^i, Y^i	Là tập hợp gồm i phần tử đầu tiên được thêm vào X và Y tương ứng
X_i, Y_i	Là trạng thái của X và Y sau vòng lặp thứ i trong vòng lặp chính; mặc định $X_0 = Y_0 = \emptyset$
O_1	O_1 là nghiệm tối ưu của bài toán SMK trên bộ dữ liệu con (f, V_1, B)
O	$O' = O_1 \setminus X_1$, $O^r = O_1 \setminus \{r\}$, với r là phần tử có chi phí lớn nhất trong O_1
O'^r	$O'^r = O' \setminus \{r\}$, với r là phần tử có chi phí lớn nhất trong O_1
$X_{<e}$, $Y_{<e}$	Với một phần tử $e \in X \cup Y$, ký hiệu $X_{<e}$, $Y_{<e}$ tương ứng là trạng thái của X, Y ngay trước thời điểm e được thêm vào X hoặc Y
θ_e^X, θ_e^Y	Với một phần tử $e \in X \cup Y$, ký hiệu θ_e^X, θ_e^Y tương ứng là trạng thái của ngưỡng θ^X, θ^Y ngay trước thời điểm e được thêm vào X hoặc Y
$l(e)$	Chỉ số vòng lặp tại đó phần tử e được chọn

Phác thảo chứng minh. Phân tích của AST dựa trên việc theo dõi hai tập ứng viên X và Y được cập nhật luân phiên. Mỗi lần một tập được cập nhật, các phần tử tối ưu không được chọn sẽ bị chặn bởi ngưỡng mật độ của vòng tương ứng, còn các phần tử được chọn ở tập kia tạo ra cận bù thông qua kỳ vọng lợi ích biên. Phần tử lớn nhất r tiếp tục được dùng để tách các trường hợp chi phí, trong khi tập V_0 gồm các phần tử rất nhỏ được xử lý bằng UnSubMax khi phù hợp. Các bổ đề lần lượt thiết lập cận cho từng nhóm phần tử, rồi định lý cộng các cận này để thu được hệ số $\frac{1}{7} - \epsilon$ cùng độ phức tạp song song $O(\log n)$.

Bổ đề 2.11. *Sau bất kỳ vòng lặp i nào của vòng lặp thứ nhất (Dòng 4-12) của AST, ta có:*

a) Nếu $i \geq 1$, i là số lẻ và $c(X_i) \leq B - c(r)$. Với $T \subseteq Y_{i-1} \cap O_1$, ta có:
$\sum_{e \in T} f(e|X_i) < \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon \cdot \mathsf{opt}$.

b) Nếu $i \geq 2$, i là số chẵn và $c(Y_i) \leq B - c(r)$. Với $T \subseteq X_{i-1} \cap O'$, ta có:
$\sum_{e \in T} f(e|Y_i) < \sum_{e \in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \cdot \mathsf{opt}$.

Bổ đề 2.11 là bước nối giữa cơ chế ngưỡng luân phiên và phân tích xấp xỉ. Ý tưởng là: khi một phần tử của nghiệm tối ưu không được chọn vào tập đang xét, mật độ lợi ích biên của nó đã bị chặn bởi ngưỡng của vòng tương ứng; trong khi đó, các phần tử được chọn vào tập còn lại cung cấp một cận bù thông qua kỳ vọng của giá trị biên.

Chứng minh. **Chứng minh a)** Nếu $i = 1$, $Y_0 = \emptyset$, Bổ đề được chứng minh. Ta xét trường hợp khác. Ta chia T thành nhiều tập con bao gồm $T = T_2 \cup T_4 \cup \ldots \cup T_{i-1}$, trong đó T_j là tập hợp tất cả các phần tử trong T được thêm vào Y_i tại vòng lặp $j \leq i - 1$. Vì $T_j \subseteq Y_{i-1} \cap O_1$ và $c(X_i) \leq B - c(r)$ nên $c(X_{<e}) + c(e) \leq c(X_i) + c(e) \leq B$ với mọi $e \in T_j$.

Do đó, ta phân loại các phần tử trong T_j thành hai tập rời nhau $T_j = T_j^1 \cup T_j^2$, trong đó $T_j^1 = \{e \in T_j : \frac{f(e|X_{<e})}{c(e)} < \theta_e^X, c(X_{<e}) + c(e) \leq B\}$ và $T_j^2 = \{e \in T_j : \frac{f(e|X_{<e})}{c(e)} \geq \theta_e^X, c(X_{<e}) + c(e) \leq B\}$. Vì $(1-\epsilon)\theta_e^X = \theta_{l(e)}^Y$ với mọi $e \in T_j^1$, ta có:

$$\begin{aligned}
\sum_{e \in T_j} f(e|X_{<e}) &= \sum_{e \in T_j^1} f(e|X_{<e}) + \sum_{e \in T_j^2} f(e|X_{<e}) \\
&< \sum_{e \in T_j^1} c(e)\theta_e^X + \frac{\mathsf{opt}}{\epsilon M} \qquad (2.23) \\
&= \sum_{e \in T_j^1} c(e)\frac{\theta_{l(e)}^Y}{1-\epsilon} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M} \qquad (2.24)
\end{aligned}$$

trong đó bất đẳng thức (2.23) theo Bổ đề 2.8, bất đẳng thức (2.24) theo Bổ đề 2.10:

$$\mathbb{E}[f(e|Y_{<e})] \geq (1-\epsilon)^2 \mathbb{E}[c(e)]\theta_{l(e)}^Y.$$

Từ đó suy ra:

$$\begin{aligned}
\sum_{e \in T} f(e|X_i) &= \sum_{j=2,4,\ldots,i-1} (\sum_{e \in T_j} f(e|X_i)) \\
&\leq \sum_{j=2,4,\ldots,i-1} (\sum_{e \in T_j} f(e|X_{<e})) \qquad (2.25) \\
&< \sum_{j=2,4,\ldots,i-1} (\sum_{e \in T_j^1} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M}) \\
&\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + (\frac{\Delta}{2} + 1) \cdot \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \sum_{e \in T} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt}, \qquad (2.26)
\end{aligned}$$

trong đó bất đẳng thức (2.25) là do tính chất submodular, bất đẳng thức (2.26) là do cách chọn M.

Chứng minh b). Nếu $i = 2$, $X_1 \cap O = \emptyset$, Bổ đề được chứng minh. Nếu $i > 2$, ta chỉ xét tập $T \subseteq X_{i-1} \cap O'$ vì ta không thể chặn $f(e|Y_{<e})$ nếu $e \in X_1$.

Ta cũng chia T thành các tập con: $T = T_3 \cup T_5 \cup \ldots \cup T_{i-1}$, trong đó T_j là tập được thêm vào X_j tại vòng lặp $j \leq i-1$. Ta cũng phân loại các phần tử trong T_j thành hai tập $T_j = T_j^1 \cup T_j^2$, trong đó $T_j^1 = \{e \in T_j : \frac{f(e|Y_{<e})}{c(e)} < \theta_{<e}^Y, c(Y_{<e}) + c(e) \leq B\}$ và $T_j^2 = \{e \in T_j : \frac{f(e|Y_{<e})}{c(e)} \geq \theta_{<e}^Y, c(Y_{<e}) + c(e) \leq B\}$. Với lập luận tương tự trường hợp

trước, ta có:

$$\begin{aligned}\sum_{e\in T_j} f(e|Y_{<e}) &= \sum_{e\in T_j^1} f(e|Y_{<e}) + \sum_{e\in T_j^2} f(e|Y_{<e}) \\ &< \sum_{e\in T_j^1} c(e)\theta_e^Y + \frac{\mathsf{opt}}{\epsilon M} \\ &= \sum_{e\in T_j^1} c(e)\frac{\theta_e^X}{1-\epsilon} + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq \sum_{e\in T_j^1} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M}\end{aligned}$$

suy ra:

$$\begin{aligned}\sum_{e\in T} f(e|Y_i) &= \sum_{j=3,5,\ldots,i-1}\sum_{e\in T_j} f(e|Y_i) \\ &\leq \sum_{j=3,5,\ldots,i-1}\sum_{e\in T_j} f(e|Y_{<e}) \\ &< \sum_{j=3,5,\ldots,i-1}(\sum_{e\in T_j^1} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M}) \\ &\leq \sum_{e\in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + (\frac{\Delta}{2}+1)\cdot\frac{\mathsf{opt}}{\epsilon M} \\ &\leq \sum_{e\in T} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon\mathsf{opt}.\end{aligned}$$

Do đó, chứng minh hoàn tất. □

Bằng việc sử dụng Bổ đề 2.11, ta tiếp tục đưa ra cận trên của $f(O' \cup T)$ với T là một tập con của X hoặc Y trong Bổ đề 2.12 khi $c(r)$ rất lớn, tức là $c(r) \geq (1-\epsilon)B$.

Bổ đề 2.12. *Nếu $c(r) \geq (1-\epsilon)B$, thì một trong hai mệnh đề sau xảy ra*

a) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5\alpha\mathsf{opt}$.

b) Tồn tại một tập con $X' \subseteq X$ sao cho

$$f(X' \cup O') < (1+\frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt}.$$

Tương tự, một trong hai mệnh đề sau xảy ra:

c) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5\alpha\mathsf{opt}$.

d) Tồn tại một tập con $Y' \subseteq Y$ sao cho

$$f(Y' \cup O') < (1+\frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt}.$$

Chứng minh. **Chứng minh a), b).** Trong trường hợp này ta có $c(X^t) \leq B - c(r) \leq \epsilon B$, ta xét các trường hợp sau:

Trường hợp 1. Nếu $X^t = X$, ký hiệu $Y_{(t)} = Y_{<x_t}$, tức là tập Y ngay trước khi thuật toán thu được X^t. Với mọi phần tử $e \in O' \setminus (X \cup Y_{(t)})$, ta có $c(X^t) + c(e) \leq B$. Do đó, ta xét $O' \setminus (X \cup Y_{(t)}) = O'^1 \cup O'^2$, trong đó $O'^1 = \{e \in O' \setminus (X \cup Y_{(t)}) : f(e|X) < c(e)\theta^X_{last}\}$, $O'^2 = \{e \in O' \setminus (X \cup Y_{(t)}) : f(e|X) \geq c(e)\theta^X_{last}\}$. Với cách đặt Δ, tại bất kỳ vòng lặp i nào ta có:

$$\frac{8\alpha f(S_0)(1-\epsilon)}{(1-8\delta)\epsilon B} = \Gamma(1-\epsilon) \geq \Gamma(1-\epsilon)^i \geq \Gamma(1-\epsilon)^\Delta \geq \frac{\epsilon f(S_0)(1-\epsilon)}{B}$$

và do đó $\theta^X_{last} \leq \frac{\Gamma(1-\epsilon)^\Delta}{1-\epsilon} \leq \frac{\epsilon \mathsf{opt}}{B}$. Áp dụng Bổ đề 2.11 và Bổ đề 2.8, ta có:

$$\begin{aligned}
f(X \cup O') - f(X) &\leq \sum_{e \in O' \setminus X} f(e|X) \quad \text{(Do tính chất submodular của } f\text{)} \\
&= \sum_{e \in O' \cap Y_{(t)}} f(e|X) + \sum_{e \in O' \setminus (X \cup Y_{(t)})} f(e|X) \\
&= \sum_{e \in O' \cap Y_{(t)}} f(e|X) + \sum_{e \in O'^1} f(e|X) + \sum_{e \in O'^2} f(e|X) \\
&\leq \sum_{e \in O' \cap Y_{(t)}} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon\mathsf{opt} + c(O'^1)\theta^X_{last} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(Y_{(t)})]}{(1-\epsilon)^3} + \epsilon\mathsf{opt} + B\theta^X_{last} + \frac{\mathsf{opt}}{\epsilon M} \\
&< \frac{\mathbb{E}[f(Y)]}{(1-\epsilon)^3} + \epsilon\mathsf{opt} + \epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

Suy ra:

$$\begin{aligned}
f(X \cup O') &\leq f(X) + \frac{\mathbb{E}[f(Y)]}{(1-\epsilon)^3} + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt} \\
&< (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt}.
\end{aligned}$$

Trường hợp 2. Nếu $X^t \neq X$, ta xét các trường hợp sau. Xét vòng lặp $z, z \geq 1$ sao cho

$$\frac{(1-\epsilon)\alpha\mathsf{opt}}{B} \leq \theta_z = \frac{\alpha\Gamma(1-\epsilon)^z}{\epsilon B} \leq \frac{\alpha\mathsf{opt}}{B}.$$

Ta định nghĩa một số nguyên lẻ l như sau:$l = \begin{cases} z, \text{nếu } z \text{ là số lẻ} \\ z+1, \text{ngược lại.} \end{cases}$

Suy ra:

$$\frac{(1-\epsilon)^2\alpha\mathsf{opt}}{B} \leq \theta_z \leq \theta^X_l \leq \theta_z \leq \frac{\alpha\mathsf{opt}}{B}.$$

Ta xét hai trường hợp sau:

Trường hợp 2.1. Nếu $X^{t+1} \subseteq X_l$, ta xét tiếp hai tình huống:

- Nếu $c(X_l) \geq (1-\epsilon)B$, theo Bổ đề 2.8 ta có:

$$\mathbb{E}[f(S)] \geq \mathbb{E}[f(X_l)] \geq \mathbb{E}[c(X_l)](1-\epsilon)^2\theta_l^X \geq (1-\epsilon)^5\alpha\mathsf{opt}.$$

- Nếu $c(X_l) < (1-\epsilon)B$. Vì $c(O^r) \leq \epsilon B$, $c(X_l) + c(e) < B$, với mọi $e \in O^r$. Kết hợp với Bổ đề 2.11 và Bổ đề 2.8, ta có:

$$\begin{aligned}
f(X_l \cup O') - f(X_l \cup \{r\}) &\leq \sum_{e \in O'^r \setminus X_l} f(e|X_l) \\
&\leq \sum_{e \in O'^r \cap Y_{l-1}} f(e|X_l) + \sum_{e \in O'^r \setminus (X_l \cup Y_{l-1})} f(e|X_l) \\
&< \sum_{e \in O'^r \cap Y_{l-1}} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon\mathsf{opt} + c(O'^r)\theta_l^X + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(Y_{l-1})]}{(1-\epsilon)^3} + \epsilon\alpha\mathsf{opt} + \alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(Y)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

Kết hợp với quy tắc chọn nghiệm cuối: $f(X_l \cup \{r\}) \leq f(S)$, ta có:

$$\begin{aligned}
f(X_l \cup O') &< f(X_l \cup \{r\}) + \frac{\mathbb{E}[f(Y)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

Trường hợp 2.2. Nếu $X_l \subset X^{t+1}$, thì $c(X_l) + c(e) \leq B$ với mọi $e \in O_1$. Do đó ta cũng có:

$$\begin{aligned}
f(X_l \cup O') - f(X_l \cup \{r\}) &\leq \sum_{e \in O'^r \setminus X_l} f(e|X_l) \\
&= \sum_{e \in O'^r \cap Y_{l-1}} f(e|X_l) + \sum_{e \in O'^r \setminus (X_l \cup Y_{l-1})} f(e|X_l) \\
&< \sum_{e \in O'^r \cap X_{l-1}} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon\mathsf{opt} + c(O'^r)\theta_l^X + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(Y_{l-1})]}{(1-\epsilon)^3} + \epsilon\alpha\mathsf{opt} + \alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(Y)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

Lưu ý $f(X_l \cup \{r\}) \leq f(S)$ do quy tắc chọn nghiệm ở vòng lặp thứ hai. Do đó:

$$f(X_l \cup O') \leq (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.$$

Kết hợp tất cả các trường hợp trên với lưu ý $\alpha < 1$, ta thu được chứng minh.

Chứng minh c), d). Ta chứng minh trường hợp này bằng lập luận tương tự như trường hợp trước và áp dụng Bổ đề 2.11 với i là số lẻ.

Trường hợp 1. Nếu $Y^u = Y$, ký hiệu $X_{(u)} = Y_{<y_u}$, tức là tập X ngay trước khi thuật

toán thu được Y^u. Với mọi phần tử $e \in O' \setminus (Y \cup X_{(u)})$, ta có $c(Y^u) + c(e) \leq B$. Do đó, ta xét $O' \setminus (Y \cup X_{(u)}) = O'_1 \cup O'_2$, trong đó $O'_1 = \{e \in O' \setminus (Y \cup X_{(u)}) : f(e|Y) < c(e)\theta^Y_{last}\}$, $O'_2 = \{e \in O' \setminus (Y \cup X_{(u)}) : f(e|Y) \geq c(e)\theta^Y_{last}\}$. Ta có: $\theta^Y_{last} \leq \frac{\Gamma(1-\epsilon)^\Delta}{1-\epsilon} \leq \frac{\epsilon \mathsf{opt}}{B}$. Áp dụng Bổ đề 2.11 và Bổ đề 2.8, ta có:

$$\begin{aligned}
f(Y \cup O') - f(Y) &\leq \sum_{e \in O' \setminus Y} f(e|Y) \quad \text{(Do tính chất submodular của } f\text{)} \\
&= \sum_{e \in O' \cap X_{(u)}} f(e|Y) + \sum_{e \in O' \setminus (Y \cup X_{(u)})} f(e|Y) \\
&= \sum_{e \in O' \cap X_{(u)}} f(e|Y) + \sum_{e \in O'_1} f(e|Y) + \sum_{e \in O'_2} f(e|Y) \\
&< \sum_{e \in O' \cap X_{(u)}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + c(O'_1)\theta^X_{last} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + B\theta^X_{last} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(X)]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + \epsilon \mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

Suy ra:

$$\begin{aligned}
f(Y \cup O') &< f(Y) + \frac{\mathbb{E}[f(X)]}{(1-\epsilon)^3} + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt} \\
&\leq (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + (2\epsilon + \frac{1}{\epsilon M})\mathsf{opt}.
\end{aligned}$$

Trường hợp 2. Nếu $Y^u \neq Y$, xét vòng lặp $z, z \geq 1$ sao cho

$$\frac{(1-\epsilon)\alpha \mathsf{opt}}{B} \leq \theta_z = \frac{\alpha\Gamma(1-\epsilon)^z}{\epsilon B} \leq \frac{\alpha \mathsf{opt}}{B}.$$

Ta định nghĩa một số nguyên chẵn l như sau: $l = \begin{cases} z, \text{nếu } z \text{ là số chẵn} \\ z+1, \text{ngược lại.} \end{cases}$

Suy ra:

$$\frac{(1-\epsilon)^2\alpha \mathsf{opt}}{B} \leq \theta_z \leq \theta^X_l \leq \theta_z \leq \frac{\alpha \mathsf{opt}}{B}.$$

Ta xét hai trường hợp sau:

Trường hợp 2.1. Nếu $Y^{u+1} \subseteq Y_l$, ta xét tiếp hai tình huống:

- Nếu $c(Y_l) \geq (1-\epsilon)B$, theo Bổ đề 2.8 ta có:

$$\mathbb{E}[f(S)] \geq \mathbb{E}[f(Y_l)] \geq \mathbb{E}[c(Y_l)](1-\epsilon)^2\theta^Y_l \geq (1-\epsilon)^5\alpha \mathsf{opt}.$$

- Nếu $c(Y_l) < (1-\epsilon)B$. Vì $c(O'^r) \leq \epsilon B$, $c(Y_l) + c(e) < B$, với mọi $e \in O'^r$. Kết hợp với Bổ đề 2.11 và Bổ đề 2.8, ta có:

$$\begin{aligned} f(Y_l \cup O') - f(Y_l \cup \{r\}) &\leq \sum_{e \in O'^r \setminus Y_l} f(e|Y_l) \\ &= \sum_{e \in O'^r \cap X_{l-1}} f(e|Y_l) + \sum_{e \in O'^r \setminus (Y_l \cup X_{l-1})} f(e|Y_l) \\ &< \sum_{e \in O'^r \cap X_{l-1}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + c(O'^r)\theta_l^Y + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq \frac{\mathbb{E}[f(X_{l-1})]}{(1-\epsilon)^3} + \epsilon\alpha\mathsf{opt} + \alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq \frac{\mathbb{E}[f(X)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}. \end{aligned}$$

Kết hợp với quy tắc chọn nghiệm cuối, ta có:

$$\begin{aligned} f(Y_l \cup O') &< f(Y_l) + \frac{\mathbb{E}[f(X)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}. \end{aligned}$$

Trường hợp 2.2. Nếu $Y_l \subset Y^{u+1}$, thì $c(Y_l) + c(e) \leq B$ với mọi $e \in O'$. Do đó ta cũng có:

$$\begin{aligned} f(Y_l \cup O') - f(Y_l \cup \{r\}) &\leq \sum_{e \in O'^r \setminus Y_l} f(e|Y_l) \\ &= \sum_{e \in O'^r \cap X_{l-1}} f(e|Y_l) + \sum_{e \in O'^r \setminus (Y_l \cup X_{l-1})} f(e|Y_l) \\ &< \sum_{e \in O'^r \cap X_{l-1}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + c(O^r)\theta_l^X + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq \frac{\mathbb{E}[f(X_{l-1})]}{(1-\epsilon)^3} + \epsilon\alpha\mathsf{opt} + \alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M} \\ &\leq \frac{\mathbb{E}[f(X)]}{(1-\epsilon)^3} + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}. \end{aligned}$$

Lưu ý: $f(Y_l \cup \{r\}) \leq f(S)$ do quy tắc chọn nghiệm ở vòng lặp thứ hai. Do đó:

$$f(Y_l \cup O') < (1 + \frac{1}{(1-\epsilon)^3})\mathbb{E}[f(S)] + 2\alpha\epsilon\mathsf{opt} + \frac{\mathsf{opt}}{\epsilon M}.$$

Kết hợp tất cả các trường hợp, ta thu được chứng minh. □

Bổ đề 2.13. *Nếu $c(X_1) < \epsilon B$ và $c(Y_2) < \epsilon B$, ta có:*

- *Nếu $c(r) < (1-\epsilon)B$, thì $f(O') \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{opt}$*
- *Nếu $c(r) \geq (1-\epsilon)B$, một trong hai trường hợp sau xảy ra:*

a) $\mathbb{E}[f(S)] \geq (1-\epsilon)^5\alpha\mathsf{opt}$.

b) $f(O') \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^3} + 2\left(2\epsilon + \frac{1}{\epsilon M}\right)\mathsf{opt}$.

Chứng minh. Chứng minh Bổ đề bằng cách xem xét cẩn thận các trường hợp sau:
Trường hợp 1. Nếu $c(r) < (1-\epsilon)B$, thì $B - c(r) > \epsilon B$.
Trường hợp 1.1. Nếu $c(X)$ và $c(Y)$ đều lớn hơn $B - c(r)$. Vì $c(X^{t+1}) > \epsilon B$ và $c(Y^{u+1}) > \epsilon B$, thuật toán phải thu được X^{t+1} và Y^{u+1} sau vòng lặp thứ i, $i \geq 3$. Ta chứng minh trường hợp này khi thuật toán thu được X^t trước Y^u. Khi xét tập con $T \subseteq O'$ trong Bổ đề 2.11, vai trò của X và Y là giống nhau. Do đó, ta có thể chứng minh tương tự cho trường hợp còn lại. Vì các phần tử trong X được thêm vào ở các vòng lẻ, nên bất kỳ phần tử $e \in O' \setminus X^t$ nào cũng không được thêm vào X tại vòng $s = l(x_{t+1}) - 2$. Rõ ràng $X_s \subseteq X^t$. Áp dụng Bổ đề 2.11, ta có:

$$\begin{aligned} f(O' \cup X^t) - f(X^t \cup \{r\}) &\leq \sum_{e \in O'^r \setminus X^t} f(e|X^t) \\ &= \sum_{e \in Y_{<x_t} \cap O'^r} f(e|X^t) + \sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) \\ &< \sum_{e \in Y_{<x_t} \cap O'^r} \frac{\mathbb{E}[f(e|Y_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + \sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t). \end{aligned} \tag{2.27}$$

Đồng thời, áp dụng Bổ đề 2.11 một lần nữa, ta có:

$$\begin{aligned} &f(O' \cup Y^u) - f(Y^u \cup \{r\}) \\ &= \sum_{e \in O'^r \cap X_{<y_u}} f(e|Y^u) + \sum_{e \in O'^r \setminus (X_{<y_u} \cup Y^u)} f(e|Y^u) \\ &\leq \sum_{e \in O'^r \cap X_{<y_u}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \epsilon \mathsf{opt} + \sum_{e \in O'^r \setminus (X_{<y_u} \cup Y^u)} f(e|Y^u). \end{aligned} \tag{2.28}$$

Do tính chất submodular của f và $X^t \cap Y^u = \emptyset$, ta có:

$$f(O') \leq f(O' \cup X^t) + f(O' \cup Y^u).$$

Từ đây, ta cần chặn các hạng tử bên phải của (2.27) và (2.28).

a) Chặn của $\sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) + \sum_{e \in O'^r \cap X_{<y_u}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3}$.
- Nếu $x_{t+1} \notin O'^r$, khi đó $O'^r \setminus (X^t \cup Y_{<x_t}) = O'^r \setminus (X^{t+1} \cup Y_{<x_t})$ và $O'^r \cap X^t = O'^r \cap X^{t+1}$. Mọi phần tử $e \in O'^r \setminus (X^{t+1} \cup Y_{<x_t})$ không được thêm vào X tại vòng $s = l(x_{t+1}) - 2$. Áp dụng Bổ đề 2.8, ta có:

$$\begin{aligned} \sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) &= \sum_{e \in O'^r \setminus (X^{t+1} \cup Y_{<x_t})} f(e|X^t) \\ &\leq \sum_{e \in O'^r \setminus (X^{t+1} \cup Y_{<x_t})} f(e|X_s) \\ &< c(O'^r \setminus (X^{t+1} \cup Y_{<x_t})) \frac{\theta^X_{(t+1)}}{(1-\epsilon)^2} + \frac{\mathsf{opt}}{\epsilon M}. \end{aligned} \tag{2.29}$$

Vì $c(X^{t+1}) > B - c(r) \geq c(O'^r)$ nên $c(X^{t+1} \setminus O'^r) > c(O'^r \setminus X^{t+1}) \geq c(O'^r \setminus (X^{t+1} \cup Y_{<x_t}))$. Kết hợp với (2.29), ta có:

$$\sum_{e \in X^{t+1} \setminus O'^r} f(e|X_{<e}) \geq c(X^{t+1} \setminus O'^r)\theta^X_{(t+1)}$$
$$\geq (1-\epsilon)^2 \Big(\sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) - \frac{\mathsf{opt}}{\epsilon M} \Big). \tag{2.30}$$

Từ (2.30) và (2.29) với lưu ý $X_{<y_u} \subseteq X^t$, ta thu được:

$$\begin{aligned}
&\sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) + \sum_{e \in O'^r \cap X_{<y_u}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} \\
&\leq \frac{\sum_{e \in X^{t+1} \setminus O'^r} f(e|X_{<e})}{(1-\epsilon)^2} + \sum_{e \in O'^r \cap X^{t+1}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M} \\
&< \frac{\mathbb{E}[f(X^{t+1})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M} \\
&\leq \frac{\mathbb{E}[f(S)]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M}.
\end{aligned}$$

- Nếu $x_{t+1} \in O'^r$, khi đó $O'^r \cap X^{t+1} = (O'^r \cap X^t) \cup \{x_{t+1}\}$ và $O'^r \setminus (X^t \cup Y^{<x_t}) = O'^r \setminus (X^{t+1} \cup Y^{<x_t}) \cup \{x_{t+1}\}$. Do đó:

$$\sum_{e \in O'^r \setminus (X^t \cup Y_{<x_t})} f(e|X^t) + \sum_{e \in O'^r \cap X_{<y_u}} \frac{\mathbb{E}[f(e|X_{<e})]}{(1-\epsilon)^3} \leq \frac{\mathbb{E}[f(X^{t+1})]}{(1-\epsilon)^3} + \frac{\mathsf{opt}}{\epsilon M}.$$

b) Chặn của $\sum_{e \in O'^r \setminus (X_{<y_u} \cup Y^u)} f(e|Y^u)$. Mọi phần tử $e \in O'^r \setminus Y^u$ không được thêm vào Y^u tại vòng $l(y_u) - 2$. Áp dụng Bổ đề 2.8 với lưu ý $c(O'^r \setminus (X_{<y_u} \cup Y^u)) \leq B - r \leq c(Y^{u+1})$, ta có:

$$\sum_{e \in O'^r \setminus (X_{<y_u} \cup Y^u)} f(e|Y^u) \leq \frac{f(Y^{u+1})}{(1-\epsilon)^4} + \frac{\mathsf{opt}}{\epsilon M}.$$

Thay các chặn trên (**a** và **b**) vào (2.27), (2.28), ta có:

$$f(O') \leq f(O' \cup X^t) + f(O' \cup Y^u) < \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{opt}.$$

Trường hợp 2: Nếu $c(r) \geq (1-\epsilon)B$. Dựa trên Bổ đề 2.12, ta xét các tình huống. Nếu một trong **a** hoặc **c** trong Bổ đề 2.12 xảy ra, thì Bổ đề 2.12 đúng. Nếu cả **b** và **d** xảy ra, ta có:

$$f(O') \leq f(O' \cup X') + f(O' \cup Y') \leq (2 + \frac{2}{(1-\epsilon)^3})\mathbb{E}[f(S)] + 2(2\epsilon + \frac{1}{\epsilon M})\mathsf{opt}$$

qua đó hoàn tất chứng minh. □

Cuối cùng, kết hợp các Bổ đề 2.11, 2.12, 2.13 và chia O thành các tập con phù hợp, ta phát biểu hiệu năng của thuật toán trong Định lý 2.5.

Định lý 2.5. *Với $\alpha = \frac{1}{7}$, $\epsilon \in (0, \frac{1}{7})$, $\delta \in (0, \frac{1}{8})$, Thuật toán 7 cần $O(\log n)$ độ phức tạp song song và $O(nk \log^2 n)$ độ phức tạp truy vấn và trả về một nghiệm S thỏa mãn* $\mathbb{E}[f(S)] \geq (\frac{1}{7} - \epsilon)\mathsf{opt}$.

Chứng minh. AST đầu tiên gọi ParSKP1 để tìm nghiệm ứng viên S_0. Bước này tốn $O(\frac{1}{\delta}\log(\frac{n}{\delta}))$ độ phức tạp song song và $O(nk\log^2 n)$ độ phức tạp truy vấn [31]. Với vòng lặp thứ nhất, thuật toán gọi RandBatch $\Delta = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ lần. Mỗi lần, RandBatch cần $O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}) + M) = O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon}) + \frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})) = O(\log n)$ độ phức tạp song song và $O(\frac{nk}{\epsilon}\log(\frac{n}{\epsilon}) + \frac{nk}{\epsilon^3}\log(\frac{1}{\epsilon})) = O(nk\log n)$ độ phức tạp truy vấn.

Trong giai đoạn thứ hai, thuật toán có thể cần $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$ độ phức tạp song song và $O(\frac{n}{\epsilon^4}\log^3(\frac{1}{\epsilon}))$ độ phức tạp truy vấn để gọi thuật toán UnSubMax của Chen và cộng sự [21] (Dòng 16-17). Sau đó, thuật toán chỉ có hai vòng lặp song song và tốn $O(kn)$ độ phức tạp truy vấn để tìm X'^i và Y'^i (Dòng 19-24). Do đó, độ phức tạp song song của thuật toán là $O(\frac{1}{\delta}\log\frac{n}{\delta}) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})\log n) + O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})) + 2 = O(\log n)$ và độ phức tạp truy vấn là $O(nk\log^2 n) + O(nk\log n) + O(\frac{n}{\epsilon^4}\log^3(\frac{1}{\epsilon})) + O(nk) = O(nk\log^2 n)$.
Về hệ số xấp xỉ, ta xét các trường hợp sau:
Trường hợp 1. Nếu $c(X_1) \geq \epsilon B$ hoặc $c(Y_2) \geq \epsilon B$, ta có

$$f(S) \geq \max\{f(X_1), f(Y_2)\} \geq \epsilon B\alpha\Gamma(1-\epsilon)^2 > \frac{(1-\epsilon)^2}{7}\mathsf{opt} > (\frac{1}{7} - \epsilon)\mathsf{opt}.$$

Trường hợp 2. Nếu $c(X_1) < \epsilon B$ và $c(Y_2) < \epsilon B$, ta có

$$f(O) \leq f(O \cap (V_1 \setminus X_1)) + f(O \cap (V_0 \cup X_1)) \tag{2.31}$$
$$= f(O') + f(O \cap (V_0 \cup X_1)) \tag{2.32}$$
$$\leq f(O') + (2+\epsilon)\mathbb{E}[f(S)] \tag{2.33}$$

trong đó bất đẳng thức (2.31) là do tính chất submodular của f và $(V_1 \setminus X_1) \cap (V_0 \cap X_1) = \emptyset$, đẳng thức (2.32) là do định nghĩa của O', và bất đẳng thức (2.33) là do áp dụng thuật toán của Chen và cộng sự [21].

Ta áp dụng Bổ đề 2.13 để chặn $f(O')$. Nếu $c(r) < (1-\epsilon)B$, thì:

$$f(O) \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{opt} + (2+\epsilon)\mathbb{E}[f(S)] \leq \frac{7\mathbb{E}[f(S)]}{(1-\epsilon)^4} + \epsilon\mathsf{opt}.$$

Do đó:

$$\mathbb{E}[f(S)] \geq \frac{(1-\epsilon)^5}{7}\mathsf{opt} > \frac{1-5\epsilon}{7}\mathsf{opt} > (\frac{1}{7} - \epsilon)\mathsf{opt}.$$

Nếu $c(r) \geq (1-\epsilon)B$, ta xét hai trường hợp:
- Nếu **a)** trong Bổ đề 2.13 xảy ra, thì

$$\mathbb{E}[f(S)] \geq \frac{(1-\epsilon)^5}{7}\mathsf{opt} > (\frac{1}{7} - \epsilon)\mathsf{opt}.$$

- Nếu **b)** trong Bổ đề 2.13 xảy ra, thì

$$f(O) \leq \frac{5\mathbb{E}[f(S)]}{(1-\epsilon)^3} + (4\epsilon + \frac{2}{\epsilon M})\mathsf{opt} + (2+\epsilon)\mathbb{E}[f(S)]$$
$$< \frac{7\mathbb{E}[f(S)]}{(1-\epsilon)^3} + \frac{29}{7}\epsilon\mathsf{opt}. \quad (2.34)$$

trong đó bất đẳng thức (2.34) là do $\epsilon M = \frac{1}{\epsilon}(\frac{\Delta}{2}+1) > \frac{14}{\epsilon}$ với $\epsilon \in (0, \frac{1}{7}), \delta \in (0, \frac{1}{8})$. Từ đó suy ra:

$$\mathbb{E}[f(S)] \geq \frac{1}{7}(1-\epsilon)^3(1-\frac{29}{7}\epsilon)\mathsf{opt} > (\frac{1}{7}-\epsilon)\mathsf{opt}.$$

Qua đó hoàn tất chứng minh. □

2.5. Thực nghiệm

Trong phần này, luận án tiến hành các thực nghiệm để đánh giá hiệu năng thực tế của ba thuật toán được đề xuất – DLA, RLA và AST – trên bài toán SMK với ba ứng dụng tiêu biểu: Tối đa lợi nhuận, Tóm tắt hình ảnh và Tối đa cắt trọng số. Do sự khác biệt về đặc trưng giữa các thuật toán song song và tuần tự, để đảm bảo tính khách quan và công bằng trong so sánh, luận án phân chia các thuật toán thành hai nhóm:

- **Nhóm thuật toán song song**: bao gồm ParSKP2, ParSKP1, ParKnapsack, AST, được đánh giá dựa trên hai tiêu chí giá trị hàm mục tiêu đạt được và độ phức tạp song song;

- **Nhóm thuật toán tuần tự**: bao gồm FANTOM, SAMPLEGREEDY, SMKDE-TACC, SMKRANACC, SMKSTREAM, RLA và DLA, được đánh giá dựa trên hai tiêu chí giá trị hàm mục tiêu và số lượng truy vấn đến hàm mục tiêu.

Cách tiếp cận này đảm bảo việc đánh giá hiệu năng giữa các thuật toán là toàn diện, minh bạch và phù hợp với đặc tính của từng nhóm.

Các thực nghiệm được thiết kế nhằm kiểm tra hai khía cạnh khác nhau của thuật toán: chất lượng nghiệm, được đo bằng giá trị hàm mục tiêu, và chi phí tính toán, được đo bằng số truy vấn oracle đối với nhóm tuần tự hoặc số vòng song song đối với nhóm song song. Việc tách hai nhóm thuật toán là cần thiết vì thuật toán tuần tự và thuật toán song song tối ưu theo hai tiêu chí vận hành khác nhau; so sánh trực tiếp số truy vấn của thuật toán tuần tự với số vòng song song của thuật toán song song sẽ không phản ánh đúng bản chất tính toán của từng nhóm.

2.5.1. Ứng dụng và bộ dữ liệu

Tối đa doanh thu. Cho một mạng xã hội được biểu diễn bởi đồ thị $G = (V, E)$, trong đó V là tập các người dùng và E là tập các kết nối giữa người dùng. Mỗi cạnh (u, v) được gán trọng số $w_{(u,v)}$ phản ánh mức độ "gần gũi" giữa u và v. Theo Mirzasoleiman

và cộng sự [90], doanh thu quảng cáo của một tập đỉnh $S \subseteq V$ được định nghĩa là:

$$f(S) = \sum_{u \in V \setminus S} \sqrt{\sum_{v \in S:(v,u) \in E} w_{(u,v)}}.$$

Trọng số $w_{(u,v)}$ được lấy ngẫu nhiên từ phân phối đều liên tục $U(0,1)$ và chi phí của mỗi đỉnh u là $c(u) = g\left(\sqrt{\sum_{(u,v) \in E} w_{(u,v)}}\right)$, với $g(x) = 1 - e^{-\mu x}$ và $\mu = 0.2$ [53]. Với ngân sách B cho trước, mục tiêu là chọn tập S có tổng chi phí không vượt quá B để tối đa $f(S)$. Đây là một trường hợp của bài toán SMK [53]. Ứng dụng này sử dụng bộ dữ liệu Facebook từ [82], gồm hơn $4,039$ đỉnh và hơn $88,234$ cạnh và bộ dữ liệu YouTube ($39,841$ nút, $224,235$ cạnh).

Tóm tắt ảnh. Cho một đồ thị $G = (V, E)$ trong đó mỗi đỉnh $u \in V$ đại diện cho một ảnh và mỗi cạnh $(u, v) \in E$ mang trọng số $w_{u,v}$ biểu thị mức độ tương đồng giữa hai ảnh. Chi phí thu thập ảnh u là $c(u)$. Mục tiêu là chọn một tập đại diện $S \subseteq V$ với ngân sách giới hạn B sao cho giá trị đại diện được tối đa:

$$f(S) = \sum_{u \in V} \max_{v \in S} w_{u,v} - \frac{1}{|V|} \sum_{u \in V} \sum_{v \in S} w_{u,v},$$

theo [53, 90]. Hàm $f(\cdot)$ là không đơn điệu, không âm và submodular [90]. Theo các công trình gần đây [53, 90], thực nghiệm được thực hiện bằng cách chọn ngẫu nhiên 500 ảnh từ tập dữ liệu CIFAR [71, 90], sau đó tính độ tương đồng giữa các ảnh dựa trên cosine similarity của các vector pixel 3,072 chiều. Cuối cùng, độ tương phản RMS được dùng làm chỉ số đánh giá chất lượng ảnh và xác định chi phí cho từng ảnh.

Tối đa cắt trọng số. Cho đồ thị $G = (V, E)$ với trọng số cạnh không âm $w_{u,v}$ cho mỗi cạnh $(u, v) \in E$. Với một tập đỉnh $S \subset V$, hàm cắt trọng số được định nghĩa là $f(S) = \sum_{u \in V \setminus S} \sum_{v \in S} w_{u,v}$. Bài toán tìm cắt trọng số lớn nhất là tìm tập $S \subseteq V$ sao cho $f(S)$ được tối đa. Theo các công trình gần đây [3, 72], hàm $f(\cdot)$ là không đơn điệu và submodular. Bộ dữ liệu được sử dụng gồm đồ thị ngẫu nhiên Erdos-Renyi với 5,000 đỉnh và xác suất cạnh là 0.2, chi phí $c(u)$ của mỗi đỉnh được chọn ngẫu nhiên từ phân phối đều $(0, 1)$ theo Amanatidis và cộng sự [3].

Bảng 2.5: Tổng hợp ứng dụng và bộ dữ liệu trong đánh giá thực nghiệm

Ứng dụng	**Bộ dữ liệu/thiết lập**	**Đặc trưng đánh giá**
Tối đa doanh thu	Facebook và YouTube; trọng số cạnh lấy từ phân phối đều; chi phí phụ thuộc trọng số kề	Đồ thị mạng xã hội, kích thước vừa và lớn
Tóm tắt ảnh	500 ảnh CIFAR; độ tương đồng cosine trên vector pixel 3,072 chiều	Dữ liệu có độ tương đồng cao, dễ xuất hiện trùng lặp thông tin
Tối đa cắt trọng số	Đồ thị Erdos-Renyi với 5,000 đỉnh, xác suất cạnh 0.2	Bài toán đồ thị mô phỏng có cấu trúc ngẫu nhiên

2.5.2. Thuật toán so sánh

Trong nhóm thuật toán tuần tự, thuật toán RLA và DLA được tiến hành so sánh với các thuật toán sau:

- FANTOM: Thuật toán ngẫu nhiên [90], có tỷ lệ xấp xỉ kỳ vọng $\frac{1}{10} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n^2}{\epsilon}\log(n))$.

- SAMPLE GREEDY (gọi tắt là GREEDY): Thuật toán ngẫu nhiên [3], có tỷ lệ xấp xỉ $\frac{1}{5.83} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))$.

- SMKDETACC: Thuật toán tất định [53], tỷ lệ xấp xỉ $\frac{1}{6} - \epsilon$, truy vấn $O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$, nhanh nhất trong các thuật toán tất định cho NSMK.

- SMKRANACC: Thuật toán ngẫu nhiên [53], tỷ lệ xấp xỉ kỳ vọng $\frac{1}{4} - \epsilon$, truy vấn $O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$.

- SMKSTREAM: Thuật toán dòng đầu tiên cho bài toán này, tỷ lệ xấp xỉ $\frac{1}{6} - \epsilon$ với truy vấn $O(\frac{n}{\epsilon}\log(B))$ [53].

Trong các thực nghiệm đối với nhóm thuật toán tuần tự, ngân sách được đặt trong khoảng từ 2% đến 12% tổng chi phí của tập cơ sở như của Amanatidis và cộng sự [3]. Tham số được đặt là $\epsilon = 0.1$ cho tất cả các thuật toán và $\alpha = \beta = \frac{1}{6}, h = 2, r = 2$ cho SMKSTREAM [53].

Trong nhóm thuật toán song song, thuật toán AST được so sánh với các thuật toán sau:

- ParSKP1: Thuật toán song song của Cui và cộng sự [30], có độ phức tạp song song $O(\log n)$ và trả về một nghiệm S thỏa mãn $\mathbb{E}[f(S)] \geq (\frac{1}{8} - \epsilon)\mathsf{opt}$.

- ParSKP2: Thuật toán Cui và cộng sự [30], có độ phức tạp song song $O(\log^2 n)$ và trả về một nghiệm S thỏa mãn $\mathbb{E}[f(S)] \geq (\frac{1}{5+2\sqrt{2}} - \epsilon)\mathsf{opt}$.

- ParKnapsack: Thuật toán song song của Amanatidis và cộng sự [2], đạt hệ số xấp xỉ $\frac{1}{9.465} - \epsilon$ trong $O(\log n)$.

Trong các thực nghiệm đối với nhóm thuật toán song song, luận án đặt tham số độ chính xác $\epsilon = 0.1$ cho tất cả các thuật toán được đánh giá và đối với AST, luận án

đặt $\delta = 0.12$. Việc lập trình song song được thực hiện bằng ngôn ngữ C++ với OpenMP. Bên cạnh đó, các thực nghiệm được tiến hành trên cụm máy chủ tính toán hiệu năng cao (HPC) với các thông số: partition=large, #threads(CPU)=128, số node=4, bộ nhớ tối đa = 3,073 GB. Đối với UnSubMax, luận án sử dụng thiết lập của các nghiên cứu trước [2, 30], cụ thể là điều chỉnh thuật toán của Feige và cộng sự [43] để đạt được tỉ lệ $\frac{1}{4} - \epsilon$ trong một vòng lặp song song và độ phức tạp truy vấn $O(n)$.

Bảng 2.6: Tổng hợp tham số và tiêu chí đánh giá trong thực nghiệm Chương 2

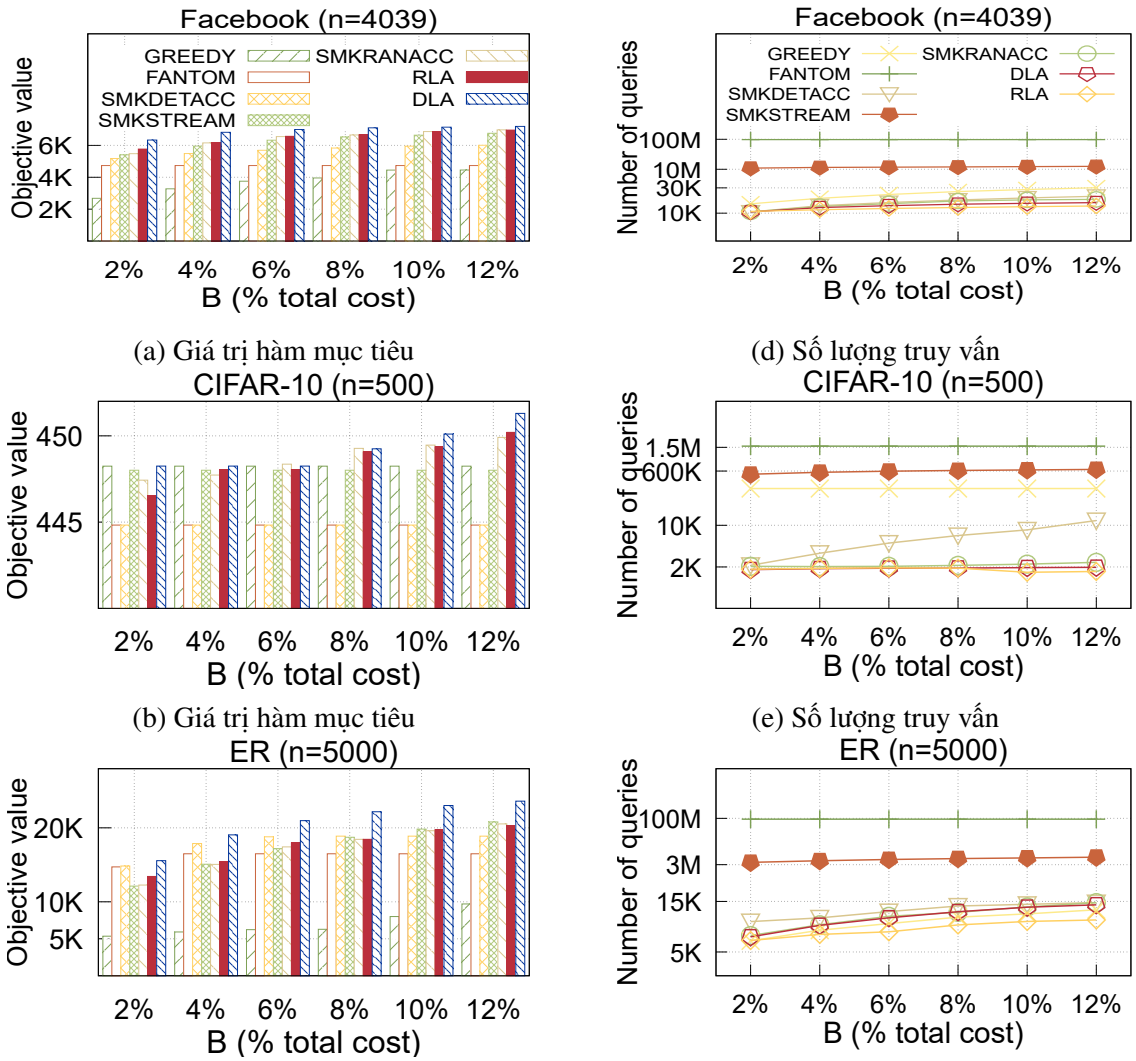
Nhóm thuật toán	**Thuật toán**	**Tham số chính**	**Tiêu chí đánh giá**
Tuần tự	FANTOM, SAMPLE-GREEDY, SMKDETACC, SMKRANACC, SMK-STREAM, DLA, RLA	$\epsilon = 0.1$; ngân sách từ 2% đến 12% tổng chi phí; SMKSTREAM dùng $\alpha = \beta = \frac{1}{6}, h = 2, r = 2$	Giá trị hàm mục tiêu và số truy vấn oracle
Song song	ParSKP2, ParSKP1, ParKnapsack, AST	$\epsilon = 0.1$; AST dùng $\delta = 0.12$; triển khai C++/OpenMP trên cụm HPC	Giá trị hàm mục tiêu và số vòng song song

2.5.3. Kết quả thực nghiệm

a) Các thuật toán tuần tự

Kết quả thực nghiệm được thể hiện trong Hình 2.6. Đầu tiên, các Hình 2.6(a), (b) và (c) minh họa chất lượng nghiệm của các thuật toán tuần tự thông qua giá trị hàm mục tiêu. Trên các thiết lập này, DLA thường đạt giá trị cao trong nhóm thuật toán tất định và cho thấy ưu thế rõ hơn so với SMKDETACC ở một số mức ngân sách. Đối với nhóm thuật toán ngẫu nhiên, RLA cho chất lượng nghiệm gần với SMKRANACC, trong khi có lợi thế về độ phức tạp truy vấn lý thuyết do không còn phụ thuộc logarit vào k. FANTOM và SAMPLEGREEDY là các baseline quan trọng vì đại diện cho những hướng tiếp cận đã được sử dụng rộng rãi, tuy nhiên trong các thực nghiệm này thường có giá trị hàm mục tiêu thấp hơn hoặc chi phí truy vấn cao hơn. Nhìn chung, kết quả cho thấy DLA và RLA đạt sự cân bằng tốt giữa chất lượng nghiệm và số truy vấn, nhưng mức độ chênh lệch phụ thuộc vào ứng dụng và ngân sách cụ thể.

Các Hình 2.6(d), (e) và (f) biểu diễn số lượng truy vấn của các thuật toán nêu trên. FANTOM là cao nhất, tiếp theo là SMKSTREAM, trong khi các thuật toán còn lại thấp hơn nhiều. FANTOM và SMKSTREAM cần hàng triệu truy vấn, trong khi các thuật toán còn lại thấp hơn hàng nghìn lần. Trong nhóm còn lại, GREEDY thường có số truy vấn cao hơn các thuật toán khác, trừ trường hợp (f). Số truy vấn của DLA, RLA và SMKRANACC là gần như tương đương, trong khi SMKDETACC dao động theo



(a) Giá trị hàm mục tiêu (d) Số lượng truy vấn

(b) Giá trị hàm mục tiêu (e) Số lượng truy vấn

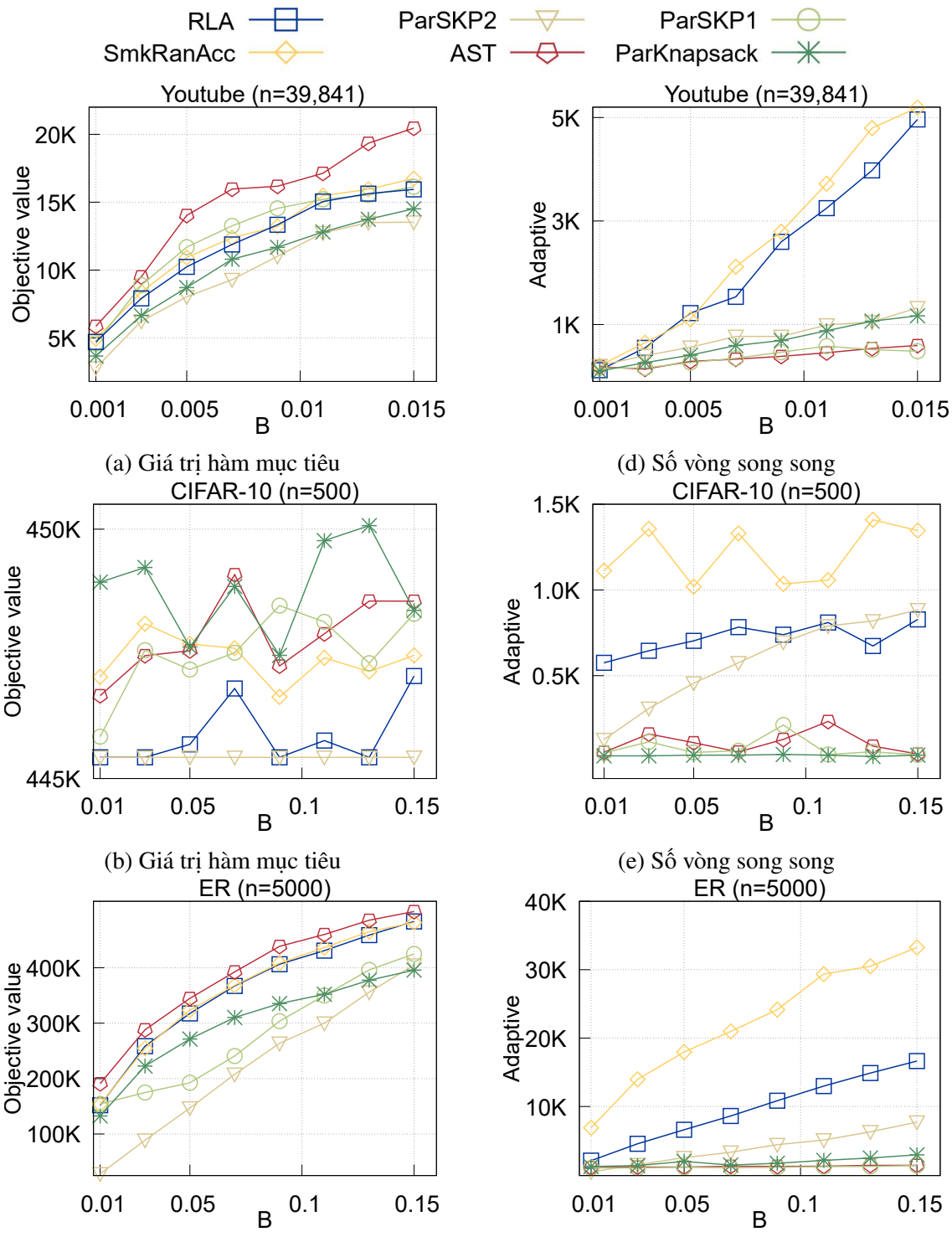
(c) Giá trị hàm mục tiêu (f) Số lượng truy vấn

- Ứng dụng: (a), (d)Tối đa doanh thu; (b), (e) Tóm tắt ảnh; và (c), (f) Cắt trọng số lớn nhất.

Hình 2.6: Kết quả thực nghiệm của các thuật toán tuần tự cho bài toán SMK.

bộ dữ liệu. RLA sử dụng ít truy vấn nhất và DLA cũng yêu cầu ít truy vấn hơn so với SMKDETACC. Khi B tăng, số lượng truy vấn của RLA tăng chậm nhất, trong khi của SMKDETACC tăng nhanh nhất. Đặc biệt, trong ứng dụng Tóm tắt ảnh, tại $B = 2\%$ tổng chi phí, các thuật toán DLA, RLA, SMKDETACC và SMKRANACC đều khoảng $2K$ truy vấn; tuy nhiên, tại $B = 12\%$, SMKDETACC cao gấp 5 lần so với các thuật toán còn lại.

Tổng thể, các thuật toán DLA và RLA của luận án thể hiện ưu điểm chính ở khả năng giữ chất lượng nghiệm cạnh tranh trong khi kiểm soát số truy vấn oracle. Hạn chế của phần thực nghiệm tuần tự là các bộ dữ liệu vẫn chủ yếu là benchmark và mô phỏng học thuật; vì vậy, các kết quả này nên được hiểu như bằng chứng thực nghiệm ban đầu về hiệu quả thuật toán, thay vì thay thế cho đánh giá trên các hệ thống triển khai thực tế.



(a) Giá trị hàm mục tiêu

(d) Số vòng song song

(b) Giá trị hàm mục tiêu

(e) Số vòng song song

(c) Giá trị hàm mục tiêu

(f) Số vòng song song

-Ứng dụng: (a), (d) Tối đa doanh thu; (b), (e) Tóm tắt hình ảnh; (c), (f) Cắt trọng số lớn nhất.

Hình 2.7: Kết quả thực nghiệm của các thuật toán song song cho bài toán SMK.

b) Các thuật toán song song

Trong Hình 2.7(a), (b) và (c) so sánh giá trị mục tiêu giữa các thuật toán khác nhau. Kết quả cho thấy AST đạt được giá trị mục tiêu tốt nhất cho cả hai ứng dụng Tối đa doanh thu và Tối đa cắt trọng số. Trong Tối đa doanh thu, các giá trị mục tiêu đạt được bởi RLA, SmkRanAcc và ParSKP1 là giống nhau, ParKnapsack đạt giá trị thấp

hơn, trong khi ParSKP2 có giá trị thấp nhất trong các thuật toán. Đặc biệt, thuật toán của luận án đạt giá trị cao nhất khi $B = 0.015$, cao hơn khoảng 1.3 lần so với các thuật toán khác trong Tối đa doanh thu. Trong Tóm tắt ảnh, ParKnapsack đạt giá trị cao nhất trong khi ParSKP2 đạt thấp nhất. Thuật toán của luận án cho kết quả tốt nhất ở một số điểm và giảm ở những điểm khác. Như thể hiện trong Hình 2.7 (b), hầu hết các thuật toán dao động mạnh. Sự biến động trong chất lượng của các thuật toán này có thể là do đặc trưng của bộ dữ liệu.

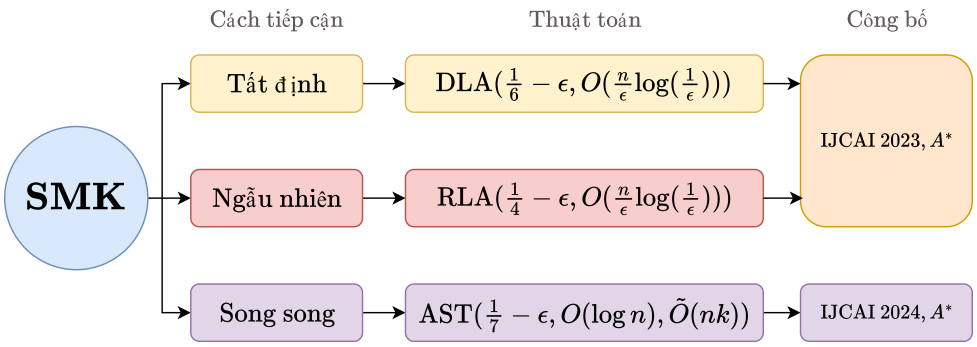
Trong Hình 2.7(d), (e) và (f), so sánh về số vòng lặp song song. Kết quả cho thấy SmkRanAcc và RLA luôn yêu cầu số vòng lặp song song cao nhất ở cả ba ứng dụng bởi vì đây là hai thuật toán tuần tự. Với AST, số vòng lặp song song tương đương với ParSKP1 trong Tối đa doanh thu và Tối đa cắt trọng số. Bên cạnh đó, trong Tóm tắt ảnh, số vòng lặp song song của AST cao hơn so với ParSKP1, vốn có số vòng thấp nhất. Tuy nhiên, sự chênh lệch này là không đáng kể.

Tổng thể, AST cho thấy lợi thế rõ ở tiêu chí số vòng song song khi so với các thuật toán tuần tự và giữ được chất lượng nghiệm cạnh tranh với các thuật toán song song liên quan. Trên bài toán Tối đa doanh thu và Tối đa cắt trọng số, AST đạt giá trị hàm mục tiêu tốt trong khi vẫn duy trì số vòng song song thấp. Trên bài toán Tóm tắt ảnh, kết quả dao động hơn và ParKnapsack có thể đạt giá trị cao hơn ở một số mức ngân sách; điều này cho thấy hiệu quả thực nghiệm của AST phụ thuộc vào cấu trúc dữ liệu và mức độ tương đồng giữa các phần tử.

2.6. Kết luận chương

Chương 2 tập trung nghiên cứu Bài toán Tối đa hàm submodular với ràng buộc chi phí (SMK) – Bài toán nghiên cứu 1 của luận án – và đề xuất ba thuật toán xấp xỉ: DLA, RLA và AST. DLA là thuật toán tất định đạt hệ số xấp xỉ $\frac{1}{6} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$, giữ cùng hệ số xấp xỉ với SMKDETACC nhưng loại bỏ sự phụ thuộc logarit vào kích thước lời giải k. RLA là thuật toán ngẫu nhiên đạt hệ số xấp xỉ kỳ vọng $\frac{1}{4} - \epsilon$ với độ phức tạp truy vấn $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon}))$, tương ứng với hướng cải thiện độ phức tạp truy vấn trong nhóm thuật toán ngẫu nhiên gần tuyến tính. AST là thuật toán song song đạt hệ số xấp xỉ kỳ vọng $\frac{1}{7} - \epsilon$, duy trì độ phức tạp song song $O(\log n)$ và độ phức tạp truy vấn $\tilde{O}(nk)$, qua đó cải thiện hệ số xấp xỉ so với thuật toán ParSKP1 trong cùng mức độ phức tạp song song. Các kết quả này cho thấy ba thuật toán được thiết kế cho các bối cảnh tính toán khác nhau: DLA và RLA phù hợp với môi trường tuần tự cần giảm số truy vấn oracle, trong khi AST phù hợp với môi trường song song. Tính đúng đắn của các thuật toán được chứng minh bằng phân tích lý thuyết và được kiểm chứng thêm bằng thực nghiệm trên các bộ dữ liệu benchmark.

Các kết quả nghiên cứu của Chương 2 đã được công bố tại hội nghị quốc tế xếp hạng **A***:



Hình 2.8: Tổng hợp các kết quả chính của luận án cho bài toán SMK.

1. Canh V. Pham, **Tan D. Tran**, Dung K.T Ha, My T. Thai, *Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint*, 2023, In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023), 4127-4135 **(RANK A*)**;

2. **Tan D. Tran**, Canh V. Pham, Dung K.T Ha, Phuong N.H. Pham, *Improved parallel algorithm for non-monotone submodular maximization under knapsack constraint*, 2024, In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024), 1961-1969 **(RANK A*)**;

CHƯƠNG 3
BÀI TOÁN TỐI ĐA HÀM k-SUBMODULAR VỚI RÀNG BUỘC CHI PHÍ NHÓM VÀ BÀI TOÁN TỐI ĐA HÀM DR-SUBMODULAR VỚI RÀNG BUỘC LỰC LƯỢNG

Trong những năm gần đây, các bài toán tối ưu hàm submodular mở rộng, bao gồm k-submodular *(Định nghĩa 1.14)* và DR-submodular *(Định nghĩa 1.16)*, đã thu hút sự quan tâm mạnh mẽ nhờ khả năng biểu diễn các bài toán có cấu trúc phức tạp trong nhiều lĩnh vực như học máy, hệ thống khuyến nghị, mạng xã hội và cảm biến thông minh. Chương này, luận án trình bày hai hướng nghiên cứu chính về hai lớp hàm này dưới các ràng buộc khác nhau. Hai hướng nghiên cứu được đặt trong cùng một chương vì cùng xuất phát từ nhu cầu mở rộng mô hình tối ưu submodular cổ điển sang các không gian quyết định giàu cấu trúc hơn. Ở hướng thứ nhất, mỗi phần tử chỉ được chọn vào một trong k nhóm và mỗi nhóm có ngân sách riêng; ở hướng thứ hai, mỗi phần tử có thể được chọn với mức độ nguyên khác nhau dưới một ràng buộc lực lượng chung. Do đó, điểm chung của chương là nghiên cứu các thuật toán xấp xỉ cho các bài toán tối ưu submodular mở rộng khi tài nguyên không còn là một ràng buộc đơn giản, mà được tổ chức theo nhóm hoặc theo mức độ lựa chọn.

Thứ nhất, bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm – Bài toán nghiên cứu 2, một mô hình phản ánh chính xác các ứng dụng như tối đa ảnh hưởng đa chủ đề và phân phối cảm biến đa loại. Để giải bài toán này trong môi trường dữ liệu lớn, luận án đề xuất một thuật toán luồng *(Định nghĩa 1.11)* mang tên STR, chỉ yêu cầu hai lượt quét qua dữ liệu và sử dụng bộ nhớ giới hạn. Thuật toán đạt hệ số xấp xỉ $\frac{1-\epsilon}{2(k+1)}$ cho hàm mục tiêu đơn điệu và $\frac{1-\epsilon}{2k+3}$ cho hàm không đơn điệu, với độ phức tạp truy vấn $O(\frac{nk}{\epsilon}\log(B))$ và bộ nhớ $O(\frac{B}{\epsilon}\log(n))$, trong đó n là kích thước dữ liệu đầu vào, k là số tập con rời nhau (theo định nghĩa k-submodular), B là tổng chi phí và $\epsilon \in (0,1)$ là tham số chính xác. Trong các thực nghiệm được trình bày ở mục sau, thuật toán này thường đạt chất lượng nghiệm cạnh tranh hoặc cao hơn phương pháp tham lam, đồng thời giảm rõ rệt số lượng truy vấn trên các bộ dữ liệu lớn. Các kết quả nghiên cứu tương ứng đã được công bố tại The 12th International Symposium on Information and Communication Technology (SOICT 2023), kỷ yếu hội thảo được lập chỉ mục trong SCOPUS và trên Asia-Pacific Journal of Operational Research (APJOR), một tạp chí quốc tế uy tín thuộc danh mục SCIE, được xếp hạng Q3.

Thứ hai, bài toán tối đa hàm DR-submodular với ràng buộc lực lượng – Bài toán nghiên cứu 3, nơi các phần tử có thể được chọn nhiều lần với mức độ khác nhau, phù hợp với các ứng dụng phân bổ tài nguyên đa cấp. Hai thuật toán được đề xuất là FastDrSubvà FastDrSub+, lần lượt đạt hệ số xấp xỉ 0.044 và $\frac{1}{4}-\epsilon$ với độ phức tạp truy vấn $O(n\log(k))$, trong đó n là kích thước dữ liệu đầu vào và k là ràng buộc lực lượng. Theo đối sánh với các hướng tiếp cận được tổng hợp trong Bảng 3.5, đây là các thuật toán xấp xỉ hằng số

có chi phí truy vấn thấp cho bài toán DrSMC trong bối cảnh hàm mục tiêu không đơn điệu. Cả hai thuật toán được kiểm chứng qua các bài toán thực nghiệm tối đa doanh thu với hàm mục tiêu dạng DR-submodular, trong đó FastDrSub+ đạt chất lượng nghiệm cạnh tranh với các thuật toán so sánh và FastDrSub có ưu thế rõ về số lượng truy vấn/thời gian chạy. Các kết quả nghiên cứu tương ứng đã được chấp nhận công bố trên Journal of Combinatorial Optimization (JOCO), một tạp chí quốc tế uy tín thuộc danh mục SCIE, được xếp hạng Q2.

Tiếp theo, luận án sẽ trình bày chi tiết mô hình bài toán, các thuật toán đề xuất, phân tích lý thuyết về hiệu quả xấp xỉ, cũng như các kết quả thực nghiệm minh họa.

3.1. Bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm

3.1.1. Định nghĩa bài toán

Hàm k-submodular *(Định nghĩa 1.14)* là một mô hình mở rộng của hàm submodular, cho phép mô hình hóa các quyết định lựa chọn khi cần phân chia một tập cơ sở V thành k tập con rời nhau. Nhờ vào tính chất tổng quát này, hàm k-submodular được sử dụng trong nhiều bài toán tối ưu rời rạc có cấu trúc phân nhóm rõ ràng.

Một trong những bài toán trung tâm là tối đa hàm k-submodular với ràng buộc chi phí, trong đó mỗi phần tử $e \in V$ đi kèm với một chi phí $c(e) > 0$ và tổng chi phí của các phần tử được chọn không vượt quá một ngân sách B. Bài toán này có thể được hiểu như việc lựa chọn một k-tập $(S_1, S_2, \ldots, S_k)$ sao cho các tập con S_i là rời nhau và tổng chi phí $\sum_{i=1}^{k} \sum_{e \in S_i} c(e) \leq B$, đồng thời giá trị của hàm mục tiêu $f(S_1, \ldots, S_k)$ được tối đa. Tuy nhiên, do ràng buộc chi phí áp dụng chung cho toàn bộ k nhóm, có thể xảy ra hiện tượng mất cân bằng nghiêm trọng: một nhóm chiếm hầu hết ngân sách trong khi các nhóm còn lại không được lựa chọn phần tử nào, dẫn đến hiệu suất tổng thể thấp hoặc không phù hợp với yêu cầu thực tế.

Để giải quyết bất cập đó, bài toán được mở rộng thành bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm, trong đó mỗi nhóm $i \in [k] = \{1, 2, \ldots, k\}$ được cấp một ngân sách riêng B_i và yêu cầu với mọi i thỏa mãn $\sum_{e \in S_i} c(e) \leq B_i$. Ràng buộc này phản ánh chính xác các yêu cầu kiểm soát độc lập ngân sách giữa các nhóm trong nhiều ứng dụng thực tiễn. Một ví dụ tiêu biểu là bài toán triển khai cảm biến k-loại (k-type sensor placement), trong đó mỗi loại cảm biến đại diện cho một nhóm và có ngân sách triển khai riêng biệt. Mục tiêu là lựa chọn vị trí đặt cảm biến sao cho độ phủ thông tin của toàn hệ thống được tối đa trong khi vẫn đảm bảo tổng chi phí triển khai của từng loại cảm biến không vượt quá giới hạn ngân sách tương ứng.

Cụ thể, Bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm (ký hiệu là kSMIK) được phát biểu như sau:

Định nghĩa 3.1 (Bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm – kSMIK)**.** Cho một tập cơ sở $V = \{e_1, e_2, \cdots, e_n\}$ gồm n phần tử, một số nguyên $k > 0$

và một hàm k-submodular $f : (k+1)^V \to \mathbb{R}_+$ xác định trên không gian họ k tập đôi một không giao nhau, được gọi là k-tập (k-set), cụ thể: $(k+1)^V = \{(S_1, S_2, \ldots, S_k) \mid S_i \subseteq V,\ \forall i \in [k],\ S_i \cap S_j = \emptyset,\ \forall i \neq j\}$. Mỗi phần tử $e \in V$ được gán một chi phí dương $c(e) > 0$. Với mỗi vị trí $i \in [k] = \{1, 2, \ldots, k\}$, tổng chi phí của tập S_i được định nghĩa là $c_i(\mathbf{s}) = \sum_{e \in S_i} c(e)$ và mỗi vị trí có một ngân sách riêng $B_i > 0$. Không mất tính tổng quát, ta giả sử với mọi $e \in V$, $c(e) \leq B$, trong đó $B = \sum_{i=1}^{k} B_i$. Mục tiêu của bài toán kSMIK là:

$$\arg \max_{\mathbf{s} \in (k+1)^V : c_i(\mathbf{s}) \leq B_i, \forall i \in [k]} f(\mathbf{s}).$$

Trong phạm vi luận án này, một thể hiện của bài toán kSMIK được ký hiệu là bộ ba (f, V, k) và nghiệm tối ưu được ký hiệu là $\mathbf{o}$, với giá trị tối ưu tương ứng là $\mathsf{opt} = f(\mathbf{o})$. Bên cạnh đó, luận án sử dụng thêm các ký hiệu sau để hỗ trợ trình bày và phân tích thuật toán:

- Ta định nghĩa $supp_i(\mathbf{s}) = S_i$, $supp(\mathbf{s}) = \cup_{i \in [k]} S_i$, trong đó S_i là tập thứ i của $\mathbf{s}$ và $\mathbf{0} = (\emptyset, \ldots, \emptyset)$ là một k-tập rỗng.

- Với $\mathbf{x} = (X_1, X_2, \ldots, X_k), \mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, ta quy ước nếu $e \in X_i$ thì $\mathbf{x}(e) = i$ và i được gọi là vị trí của e, ngược lại $\mathbf{x}(e) = 0$. Việc thêm một phần tử $e \notin supp(\mathbf{x})$ vào X_i được biểu diễn bởi $\mathbf{x} \sqcup (e, i)$. Khi $X_i = \{e\}$ và $X_j = \emptyset$, với mọi $j \neq i$, $\mathbf{x}$ được ký hiệu bởi (e, i). Ta ký hiệu $\mathbf{x} \sqsubseteq \mathbf{y}$ khi và chỉ khi $X_i \subseteq Y_i$ với mọi $i \in [k]$.

- Lợi ích biên của (e, i) khi được thêm vào $\mathbf{x} \in (k+1)^V$ được ký hiệu là $\Delta_{(e,i)} f(\mathbf{x}) = f(\mathbf{x} \sqcup (e, i)) - f(\mathbf{x})$.

Bài toán kSMIK có tính ứng dụng cao trong các hệ thống ra quyết định đa mục tiêu, nơi tài nguyên phải được phân bổ đồng thời cho nhiều nhóm độc lập, mỗi nhóm đi kèm với một ràng buộc ngân sách riêng. Mô hình này xuất hiện phổ biến trong các lĩnh vực như tối ưu phân phối nội dung, quản lý tài nguyên trong mạng cảm biến, thiết kế chiến lược tiếp thị đa kênh và phân bổ ngân sách trong các hệ thống khuyến nghị cá nhân hóa. Trong lĩnh vực học máy, kSMIK có thể được ứng dụng để chọn tập huấn luyện đại diện theo nhiều chủ đề hoặc nhóm người dùng khác nhau, mỗi nhóm có giới hạn riêng về dữ liệu hoặc chi phí thu thập, với mục tiêu tối đa hiệu quả học tổng thể. Ngoài ra, bài toán còn phù hợp với các kịch bản chọn đặc trưng đa nhóm hoặc tối ưu hoá mô hình trong môi trường đa tác vụ, nơi các ràng buộc tài nguyên cần được kiểm soát tách biệt theo từng chiều chiến lược hoặc nhóm tác vụ cụ thể.

3.1.2. Nghiên cứu liên quan

Nhằm làm rõ động lực nghiên cứu, luận án tổng hợp và phân tích các công trình liên quan đến tối ưu hàm k-submodular theo hai hướng chính: (1) bài toán tối đa hàm k-submodular dưới nhiều loại ràng buộc và (2) các thuật toán luồng áp dụng cho hàm

k-submodular. Việc hệ thống hóa này không chỉ giúp xác định khoảng trống trong nghiên cứu hiện tại mà còn là cơ sở để xây dựng và phát triển các thuật toán mới phù hợp với bài toán đặt ra trong luận án.

Tối ưu hàm k-submodular. Bài toán tối ưu k-submodular được khởi nguồn từ công trình của Singh và cộng sự [111], trong đó tác giả giải quyết trường hợp $k = 2$, còn gọi là tối đa hàm bisubmodular. Sau đó, bài toán được mở rộng cho mọi giá trị k. Đáng chú ý, khi $k = 1$, bài toán trở thành bài toán tối đa hàm submodular cổ điển, vốn đã được chứng minh là NP-khó. Điều này kéo theo bài toán k-submodular cũng thuộc lớp bài toán NP-khó. Đối với trường hợp không ràng buộc, Ward và cộng sự [123] đề xuất một thuật toán tham lam mang tính quyết định với hệ số xấp xỉ $\frac{1}{3}$. Tiếp theo, Iwata và cộng sự [63] cải tiến bằng một phiên bản tham lam ngẫu nhiên, đạt hệ số xấp xỉ $\frac{k}{2k-1}$ thông qua việc kết hợp phân phối xác suất với cách lựa chọn các phần tử có lợi ích biên lớn. Sau đó, Oshima và cộng sự [99] loại bỏ tính ngẫu nhiên trong thuật toán, tuy nhiên điều này làm tăng số lượng truy vấn cần thiết. Các bài toán k-submodular có ràng buộc cũng đã được nghiên cứu ở nhiều hướng. Ohsaka và cộng sự [98] tập trung vào bài toán tối đa hàm k-submodular đơn điệu dưới ràng buộc lực lượng và đề xuất một thuật toán tham lam đạt hệ số xấp xỉ $\frac{1}{2}$ trong trường hợp ràng buộc tổng lực lượng và $\frac{1}{3}$ trong trường hợp đơn lực lượng. Qian và cộng sự [104] giới thiệu một thuật toán tiến hóa đa mục tiêu cho bài toán tối đa k-submodular đơn điệu dưới ràng buộc tổng kích thước. Trong khi đó, Zheng và cộng sự [127] nghiên cứu bài toán với hàm mục tiêu được xấp xỉ bởi một hàm k-submodular gần đúng, nhằm phục vụ cho các bài toán có ràng buộc về kích thước. Ngoài ra, bài toán tối ưu k-submodular dưới các ràng buộc phức tạp khác cũng đã được quan tâm. Sakaue [108] trình bày một thuật toán tham lam với hệ số xấp xỉ $\frac{1}{2}$ cho bài toán có ràng buộc cấu trúc. Tiếp đó, Rafiey và cộng sự [105] đề xuất một phương pháp tham lam liên tục riêng biệt, cũng đạt hệ số xấp xỉ $\frac{1}{2}$ cho cùng bài toán. Gần đây, nhiều nghiên cứu đã tập trung vào bài toán k-submodular dưới ràng buộc chi phí [51, 102, 103, 118, 122] và đạt được các hệ số xấp xỉ hằng số đáng kể.

Thuật toán luồng cho hàm k-submodular. Đối với các hàm k-submodular, [63, 123] đã phát triển các thuật toán luồng *(Định nghĩa 1.11)* một lượt quét. Trong đó, Iwata và cộng sự [63] áp dụng lựa chọn ngẫu nhiên và các nghiên cứu tiếp theo đã cải tiến bằng cách sử dụng phân phối chọn phần tử mới dựa trên chi phí khác nhau và thiết lập mối quan hệ giữa nghiệm hiện tại và nghiệm tối ưu. Rafiey và cộng sự [105] là những người đầu tiên đề xuất thuật toán luồng cho bài toán tối đa k-submodular dưới ràng buộc tổng kích thước trong điều kiện có nhiễu. Họ đưa ra hai thuật toán với hệ số xấp xỉ $O(\frac{\epsilon B}{(1-\epsilon)^2})$ cho hàm đơn điệu và $O(\frac{\epsilon B}{(1-\epsilon)^3})$ cho hàm không đơn điệu. Gần đây, Phạm và cộng sự [103] đề xuất hai thuật toán luồng một lượt quét cho bài toán tối đa k-submodular dưới ràng buộc chi phí, với độ phức tạp $O(\frac{nk}{\epsilon}\log(n))$ và hệ số xấp xỉ kỳ vọng lần lượt là $\frac{1}{4} - \epsilon$ và $\frac{k}{4k-1} - \epsilon$.

Theo rà soát các công trình liên quan nêu trên, các nghiên cứu hiện có đã xem xét nhiều dạng ràng buộc cho bài toán k-submodular, nhưng chưa tập trung trực tiếp vào ràng buộc chi phí riêng cho từng nhóm như trong bài toán kSMIK. Điều này chỉ ra một khoảng trống nghiên cứu đáng kể và nhấn mạnh sự cần thiết phải phát triển các phương pháp và thuật toán mới. Việc lấp đầy khoảng trống nghiên cứu này sẽ góp phần quan trọng vào việc hoàn thiện lý thuyết tối ưu k-submodular và ứng dụng nó vào các bài toán thực tiễn có ràng buộc chi phí nhóm.

3.1.3. Thuật toán đề xuất

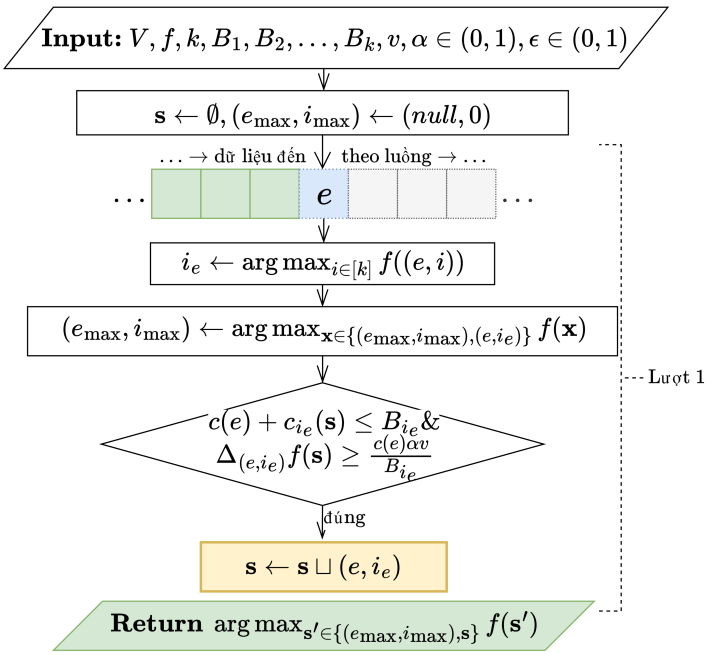
Trong mục này, luận án trình bày một thuật toán luồng để giải bài toán kSMIK. Trước tiên, một phiên bản đơn giản của thuật toán được giới thiệu với giả định biết trước giá trị tối ưu opt, gọi là STROPT. Thuật toán này đóng vai trò nền tảng để xây dựng thuật toán chính STR, không yêu cầu biết trước opt. Thuật toán STR có độ phức tạp truy vấn là $O(\frac{Bk}{\epsilon}\log(n))$, sử dụng bộ nhớ $O(\frac{B}{\epsilon}\log(n))$ và đạt hệ số xấp xỉ $\frac{1-\epsilon}{2(k+1)}$ đối với hàm mục tiêu đơn điệu và $\frac{1-\epsilon}{2k+3}$ đối với hàm không đơn điệu. Các chi tiết về thuật toán cùng phân tích lý thuyết tương ứng sẽ được trình bày trong các mục sau.

3.1.3.1. Thuật toán luồng STROPT

Thuật toán STROPT được xây dựng dựa trên ý tưởng sử dụng một giá trị xấp xỉ v của giá trị tối ưu opt như một ngưỡng để hướng dẫn quá trình lựa chọn phần tử vào tập lời giải. Cụ thể, thuật toán đánh giá mỗi phần tử từ luồng dữ liệu dựa trên hai tiêu chí: giá trị lợi ích biên khi thêm phần tử đó vào một trong các tập con hiện tại và tuân thủ ràng buộc ngân sách. Bên cạnh đó, để tránh trường hợp không chọn được phần tử nào trong toàn bộ luồng, thuật toán luôn duy trì và cập nhật một cặp phần tử đơn $(e_{\max}, i_{\max})$ có giá trị mục tiêu cao nhất quan sát được tính đến thời điểm hiện tại.

Thuật toán nhận đầu vào là một thể hiện (f, V, k) của bài toán, cùng với giá trị xấp xỉ v sao cho $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$, bộ ngân sách $(B_1, B_2, \ldots, B_k)$ và các tham số điều khiển $\alpha, \epsilon \in (0,1)$, trong đó opt là giá trị tối ưu của bài toán. Quá trình xử lý dữ liệu được thực hiện trong một lượt quét duy nhất, với mỗi phần tử e lần lượt được xét đến theo thứ tự xuất hiện.

Tại mỗi bước, thuật toán xác định vị trí i_e trong số k tập con sao cho phần tử e mang lại lợi ích biên lớn nhất khi được thêm vào vị trí đó. Đồng thời, thuật toán so sánh giá trị $f((e, i_e))$ với giá trị cao nhất hiện có và cập nhật cặp $(e_{\max}, i_{\max})$ nếu cần. Nếu phần tử e thỏa mãn cả hai điều kiện: (i) không vượt quá ngân sách nhóm tại vị trí i_e và (ii) lợi ích biên đủ lớn so với ngưỡng $\frac{\alpha v}{B_{i_e}}$, thì cặp (e, i_e) sẽ được thêm vào tập ứng viên $\mathbf{s}$.



Hình 3.1: Lưu đồ hoạt động của thuật toán luồng STROPT

Sau khi kết thúc quá trình quét, thuật toán trả về nghiệm tốt nhất giữa tập ứng viên $\mathbf{s}$ và phần tử đơn $(e_{\max}, i_{\max})$. Cụ thể, các bước được trình bày trong Thuật toán 8 và được minh họa trong Hình 3.1.

Algorithm 8 Thuật toán luồng STROPT

Input: V, f, k, $B_1, B_2, \ldots, B_k$, an approximate value v of opt such that $(1-\epsilon)\text{opt} \leq v \leq \text{opt}$, parameters $\alpha, \epsilon \in (0,1)$.
Output: A solution $\mathbf{s}$ of kSMIK.
$\mathbf{s} \leftarrow \emptyset$, $(e_{max}, i_{max}) \leftarrow (null, 0)$
for all $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e_{max}, i_{max}), (e, i_e)\}} f(\mathbf{x})$
  if $c(e) + c_{i_e}(\mathbf{s}) \leq B_{i_e}$ **then**
    if $\Delta_{(e,i_e)} f(\mathbf{s}) \geq \frac{c(e)\alpha v}{B_{i_e}}$ **then**
      $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
    end if
  end if
end for
$\mathbf{s} \leftarrow \arg\max_{\mathbf{s}' \in \{(e_{max}, i_{max}), \mathbf{s}\}} f(\mathbf{s}')$
return $\mathbf{s}$

Bảng 3.1: Bảng các ký tự toán học dùng trong phân tích thuật toán STROPT

Ký hiệu	Ý nghĩa
$\mathbf{o}$	Nghiệm tối ưu của bài toán trên tập V, với giá trị tối ưu tương ứng là $\mathsf{opt} = f(\mathbf{o})$
(e_j, i_j)	Cặp phần tử thứ j được thêm vào trong vòng lặp chính của Thuật toán 8
t	Số lượng cặp phần tử được thêm vào tập $\mathbf{s}$ sau khi kết thúc vòng lặp chính
$\mathbf{s}^t$	$\mathbf{s}^t = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: k-tập $\mathbf{s}$ sau khi vòng lặp chính kết thúc, với $t = \lvert supp(\mathbf{s}^t)\rvert$
$\mathbf{s}^j$	$\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: k-tập $\mathbf{s}$ tại bước thứ j trong vòng lặp chính, với $1 \leq j \leq t$; ban đầu $\mathbf{s}^0 = \mathbf{0}$ và sau cùng $\mathbf{s}^t = \mathbf{s}$
$\mathbf{o}^j$	$\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$
v	Giá trị dự đoán của opt, thoả mãn: $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$
$\mathbf{o}^{j-1/2}$	$\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$
$\mathbf{s}^{j-1/2}$	$\mathbf{s}^{j-1/2}$: Nếu $e_j \in supp(\mathbf{o})$, thì $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}(e_j))$. Nếu $e_j \notin supp(\mathbf{o})$, thì $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$

Tiếp theo, luận án tiến hành phân tích hệ số của thuật toán STROPT. Trước tiên, một số ký hiệu quan trọng được sử dụng trong quá trình phân tích được được định nghĩa hoặc nhắc lại tại Bảng 3.1.

Phác thảo chứng minh. Phân tích của STROPT dựa trên việc so sánh nghiệm đang xây dựng $\mathbf{s}^j$ với nghiệm tối ưu $\mathbf{o}$ sau từng phần tử được chấp nhận từ luồng dữ liệu. Bổ đề đầu tiên cho thấy phần giá trị tối ưu bị mất khi thay dần $\mathbf{o}$ bởi các phần tử của $\mathbf{s}^j$ có thể được chặn bởi giá trị của $\mathbf{s}^j$; hệ số chặn khác nhau giữa trường hợp đơn điệu và không đơn điệu. Các bổ đề tiếp theo tách hai khả năng: nếu nhiều phần tử tối ưu không được chọn vì không đủ lợi ích biên thì tổng đóng góp còn lại bị chặn bởi ngưỡng; nếu không được chọn vì hết ngân sách nhóm thì nghiệm đã sử dụng đủ ngân sách để bảo đảm một cận dưới cho giá trị. Kết hợp hai tình huống này và chọn tham số α phù hợp cho ra hệ số xấp xỉ của thuật toán.

Trước hết, luận án thiết lập mối quan hệ giữa $f(\mathbf{o})$ và $f(\mathbf{s}^j)$ thông qua Bổ đề 3.1.

Bổ đề 3.1. *Với mọi j, $0 \leq j \leq t$, ta có:*

- *Nếu f là hàm đơn điệu, thì $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq f(\boldsymbol{s}^j)$.*
- *Nếu f là hàm không đơn điệu, thì $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq 2f(\boldsymbol{s}^j)$.*

Chứng minh. Xét trường hợp f là hàm đơn điệu. Ta có $\mathbf{o}^0 = (\mathbf{o} \sqcup \mathbf{s}^0) \sqcup \mathbf{s}^0$ và do đó $f(\mathbf{o}) = f(\mathbf{o}^0)$. Với mọi $0 \leq j \leq t$, ta có:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{o}^j) &= \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^i)) \\ &\leq \sum_{i=1}^{j} (f(\mathbf{o}^{i-1}) - f(\mathbf{o}^{i-1/2})) \text{ (theo tính đơn điệu của} f) \end{aligned}$$

$$\leq \sum_{i=1}^{j}(f(\mathbf{s}^{i-1/2}) - f(\mathbf{s}^{i-1})) \text{ (hệ quả tính chất } k\text{-submodular)}$$
$$\leq \sum_{i=1}^{j}(f(\mathbf{s}^{i}) - f(\mathbf{s}^{i-1})) \text{ (theo cách chọn của thuật toán)}$$
$$\leq f(\mathbf{s}^{j}).$$

Xét trường hợp f là hàm không đơn điệu. Ta cần sử dụng tính chất đơn điệu theo từng cặp (pairwise-monotonicity) để phân tích chặt chẽ hiệu $f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j})$. Nhắc lại (e_j, i_j) là cặp phần tử thứ j được thêm vào nghiệm ứng viên $\mathbf{s}$ sau vòng lặp của Thuật toán 8. Ta xét hai trường hợp:

Trường hợp 1: Nếu $e_j \notin supp(\mathbf{o})$, đặt $l \in [k]$, $l \neq i_j$ và $\mathbf{o}_l^j$ là k-tập thỏa mãn $\mathbf{o}_l^j(e) = \mathbf{o}^j(e)$ với mọi $e \in V \setminus \{e_j\}$ và $\mathbf{o}_l^j(e_j) = l$, ta có:

$$\begin{aligned} f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j}) &= f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) - (f(\mathbf{o}^j) + f(\mathbf{o}_l^j) - 2f(\mathbf{o}^{j-1})) \\ &\leq f(\mathbf{o}_l^j) - f(\mathbf{o}^{j-1}) \text{ (theo tính đơn điệu từng cặp)} \\ &\leq f(\mathbf{s}_l^j) - f(\mathbf{s}^{j-1}) \text{ (hệ quả tính chất } k\text{-submodular)} \\ &\leq f(\mathbf{s}^j) - f(\mathbf{s}^{j-1}) \text{ (theo cách chọn của thuật toán).} \end{aligned}$$

Trường hợp 2: Nếu $e_j \in supp(\mathbf{o})$, ta xét hai khả năng:

- Nếu $\mathbf{o}^{j-1}(e_j) = i_j$, do tính đơn điệu theo từng cặp của f, tồn tại $i' \in [k]$ sao cho $f(\mathbf{s}^{j-1} \sqcup (e_j, i')) \geq 0$. Khi đó, $f(\mathbf{o}^j) - f(\mathbf{o}^{j-1}) = 0 \leq f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})$.
- Nếu $\mathbf{o}^{j-1}(e_j) \neq i_j$, ta có:

$$\begin{aligned} f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j}) &= 2f(\mathbf{o}^{j-1}) - 2f(\mathbf{o}^{j-1/2}) - (f(\mathbf{o}^{j-1}) + f(\mathbf{o}^j) - 2f(\mathbf{o}^{j-1/2})) \\ &\leq 2f(\mathbf{o}^{j-1}) - 2f(\mathbf{o}^{j-1/2}) \\ &\leq 2f(\mathbf{s}^j) - 2f(\mathbf{s}^{j-1}). \end{aligned}$$

Tổng hợp hai trường hợp trên, ta thu được: $f(\mathbf{o}^{j-1}) - f(\mathbf{o}^{j}) \leq 2f(\mathbf{s}^j) - 2f(\mathbf{s}^{j-1})$. Suy ra:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{o}^t) &= \sum_{j=1}^{t}(f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)) \\ &\leq 2\sum_{j=1}^{t}(f(\mathbf{s}^j) - f(\mathbf{s}^{j-1})) \\ &\leq 2f(\mathbf{s}^t). \end{aligned}$$

Vậy chứng minh hoàn tất. □

Bổ đề 3.2. *Sau khi kết thúc vòng lặp đầu tiên, nếu với mọi cặp (e, i) thỏa mãn $e \in supp(\boldsymbol{o}) \setminus supp(\boldsymbol{s}^t)$ và $i \in [k]$ sao cho $c_i(\boldsymbol{s}^t) + c(e) > B_i$ và $\Delta_{(e,i)}f(\boldsymbol{s}^t) < \frac{c(e)\alpha v}{B_i}$, thì:*

- *Nếu f là hàm đơn điệu, ta có $f(\boldsymbol{s}^t) > \frac{v(1-\alpha k)}{2}$.*

- *Nếu f là hàm không đơn điệu, ta có $f(\boldsymbol{s}^t) > \frac{v(1-\alpha k)}{3}$.*

Chứng minh. Với trường hợp f là hàm đơn điệu, ta có:

$$\begin{aligned} f(\mathbf{o}^t) - f(\mathbf{s}^t) &\leq \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^t)} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \text{ (hệ quả tính chất } k\text{-submodular)} \\ &= \sum_{i=1}^{k} \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}),\ \mathbf{o}(e)=i} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \\ &< \sum_{i=1}^{k} \frac{c_i(\mathbf{o})\alpha v}{B_i} \text{ (giả thiết trong bổ đề)} \\ &\leq \sum_{i=1}^{k} \frac{B_i \alpha v}{B_i} = k\alpha v. \end{aligned}$$

Mặt khác, theo Bổ đề 3.1, ta có:

$$\begin{aligned} v - f(\mathbf{s}^t) &\leq f(\mathbf{o}) - f(\mathbf{s}^t) \\ &= f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \\ &< f(\mathbf{s}^t) + k\alpha v. \end{aligned}$$

Suy ra $f(\mathbf{s}^t) \geq \frac{1}{2} v(1 - \alpha k)$.

Xét trường hợp f là hàm không đơn điệu. Ta áp dụng lập luận tương tự với trường hợp đơn điệu:

$$\begin{aligned} v - f(\mathbf{s}^t) &\leq f(\mathbf{o}) - f(\mathbf{s}^t) \\ &= f(\mathbf{o}) - f(\mathbf{o}^t) + f(\mathbf{o}^t) - f(\mathbf{s}^t) \text{ (theo Bổ đề } 3.1) \\ &\leq 2f(\mathbf{s}^t) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \\ &= 2f(\mathbf{s}^t) + \sum_{i=1}^{k} \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}),\ \mathbf{o}(e)=i} \Delta_{(e,\mathbf{o}(e))} f(\mathbf{s}^t) \\ &< 2f(\mathbf{s}^t) + \sum_{i=1}^{k} \frac{B_i \alpha v}{B_i} \text{ (theo cách chọn của thuật toán)} \\ &= 2f(\mathbf{s}^t) + k\alpha v. \end{aligned}$$

Do đó, ta có $f(\mathbf{s}^t) > \frac{1}{3} v(1 - \alpha k)$. Vậy chứng minh hoàn tất. □

Bổ đề 3.3. *Nếu tồn tại một phần tử $e \in supp(\boldsymbol{o}) \setminus supp(\boldsymbol{s}^t)$ và một số nguyên $i \in [k]$ sao cho $c_i(\boldsymbol{s}^t) + c(e) \geq B_i$ và $\Delta_{(e,i)} f(\boldsymbol{s}^t) > \frac{c(e)\alpha v}{B_i}$, thì $f(\boldsymbol{s}) \geq \frac{\alpha v}{2}$.*

Chứng minh. Vì $\Delta_{(e,i)} f(\mathbf{s}^t) \geq c(e)\alpha \frac{v}{B_i}$, nên ta có:

$$f(\mathbf{s} \sqcup (e,i)) - f(\mathbf{s}^t) \geq \frac{c(e)\alpha v}{B_i}$$

$$
\begin{aligned}
\implies f(\mathbf{s}^t \sqcup (e,i)) &\geq f(\mathbf{s}^t) + \frac{c(e)\alpha v}{B_i} \\
&\geq \frac{c_i(\mathbf{s}^t)}{B_i} + \frac{c(e)\alpha v}{B_i} \qquad (3.1) \\
&\geq \frac{(c_i(\mathbf{s}^t) + c(e))\alpha v}{B_i} \\
&\geq \alpha v. \qquad (3.2)
\end{aligned}
$$

Trong đó, bất đẳng thức (3.1) là do quy tắc lựa chọn của thuật toán. Lưu ý, sau khi vòng lặp chính của Thuật toán 8 kết thúc, ta có $(e_{max}, i_{max}) = \arg\max_{e \in V, i \in [k]} f((e,i))$, do đó:

$$
\begin{aligned}
f(\mathbf{s}) &\geq \max\{f(\mathbf{s}^t), f((e_{max}, i_{max}))\} \\
&\geq \frac{f(\mathbf{s}^t) + f((e_{max}, i_{max}))}{2} \\
&\geq \frac{f(\mathbf{s}^t) + f((e,i))}{2} \\
&\geq \frac{f(\mathbf{s}^t \sqcup (e,i))}{2} \\
&\geq \frac{\alpha v}{2}. \qquad (3.3)
\end{aligned}
$$

Bất đẳng thức (3.3) là hệ quả trực tiếp của (3.2). Chứng minh hoàn tất. □

Cuối cùng, luận án phát biểu kết quả hiệu năng của Thuật toán 8 như sau:

Định lý 3.1. *Với $\epsilon, \alpha \in (0,1)$ và giá trị xấp xỉ v sao cho $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$, Thuật toán 8 có độ phức tạp truy vấn là $O(nk)$ và:*

- Nếu f là hàm đơn điệu và $\alpha = \frac{1}{k+1}$, thì thuật toán trả về nghiệm xấp xỉ với tỉ lệ $\frac{1-\epsilon}{2(k+1)}$.

- Nếu f là hàm không đơn điệu và $\alpha = \frac{2}{2k+3}$, thì thuật toán trả về nghiệm xấp xỉ với tỉ lệ $\frac{1-\epsilon}{2k+3}$.

Chứng minh. Thuật toán thực hiện tối đa n vòng lặp. Ở mỗi vòng lặp, thuật toán thực hiện tối đa k truy vấn để quyết định có thêm phần tử e vào $\mathbf{s}$ hay không. Do đó, tổng độ phức tạp truy vấn là $O(nk)$.

Tiếp theo, ta chứng minh giới hạn dưới cho giá trị lời giải. Với trường hợp f là hàm đơn điệu, áp dụng các Bổ đề 3.2 và 3.3, ta xét hai trường hợp:

Trường hợp 1. Sau khi kết thúc vòng lặp chính, nếu với mọi $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})$ và $i \in [k]$ sao cho $c_i(\mathbf{s}) + c(e) > B_i$, thì $\Delta_{(e,i)} f(\mathbf{s}) < c(e)\alpha v$. Theo Bổ đề 3.2 và chọn $\alpha = \dfrac{1}{k+1}$, ta được:

$$
f(\mathbf{s}) > \frac{v(1-\alpha k)}{2} \geq \frac{v}{2(k+1)} = \frac{\mathsf{opt}(1-\epsilon)}{2(k+1)}. \qquad (3.4)
$$

Trường hợp 2. Nếu tồn tại $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s})$ và $i \in [k]$ sao cho $c_i(\mathbf{s}) + c(e) > B_i$ và $\Delta_{(e,i)} f(\mathbf{s}) \geq c(e)\alpha v$, thì theo Bổ đề 3.3, ta cũng có:

$$f(\mathbf{s}) \geq f(\mathbf{s}^t) \geq \frac{v}{2(k+1)} \geq \frac{\mathsf{opt}(1-\epsilon)}{2(k+1)}.$$

Với trường hợp f là hàm không đơn điệu, ta áp dụng lập luận tương tự như với hàm đơn điệu để suy ra kết quả. □

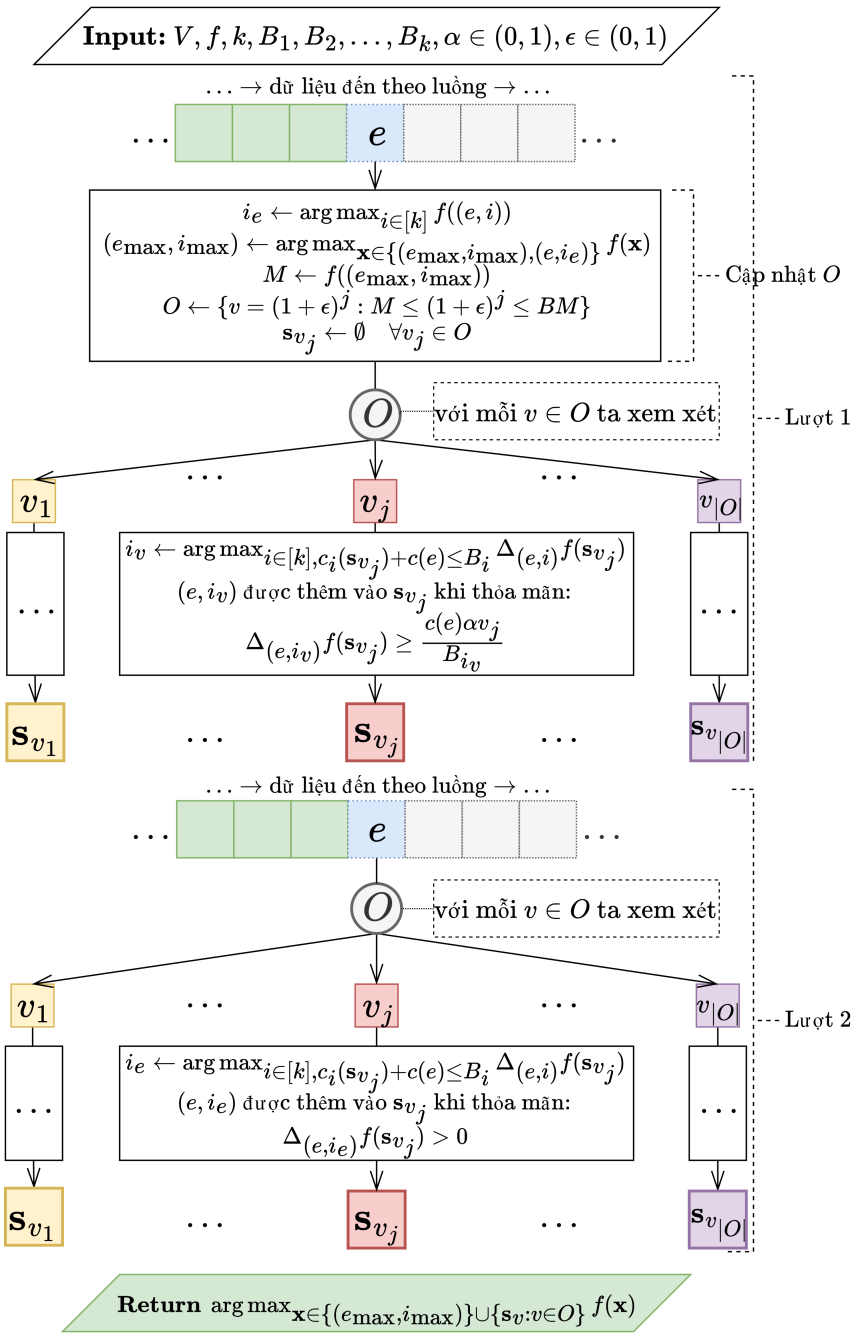
3.1.3.2. Thuật toán luồng STR

Sau khi đã trình bày thuật toán đầu tiên với giả định biết trước giá trị tối ưu opt, luận án tiếp tục giới thiệu thuật toán chính mang tên STR (Thuật toán 9), trong đó giả định về opt được loại bỏ. STR khởi tạo một tập các giá trị ước lượng cho opt theo công thức $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$, trong đó $B = \sum_{i=1}^{k} B_i$ và M là giá trị lớn nhất của các phần tử đơn lẻ (e_{max}, i_{max}) được gặp cho đến thời điểm hiện tại. Thuật toán STR bao gồm hai lượt quét dữ liệu:

Trong lượt thứ nhất (dòng 4–15), thuật toán xây dựng một tập các nghiệm ứng viên $\mathbf{s}_v$ tương ứng với mỗi giá trị $v \in O$. Với mỗi phần tử đến e, thuật toán thực hiện hai thao tác:

- Cập nhật giá trị M và tập O dựa trên các phần tử đã nhận (dòng 7–8). Cụ thể, M sẽ được cập nhật theo giá trị tốt nhất hiện tại của cặp đơn lẻ (e_{max}, i_{max}) và từ đó, tập O được cập nhật lại tương ứng.
- Thêm phần tử e vào các nghiệm ứng viên $\mathbf{s}_v$ với $v \in O$, nếu phần tử này có lợi ích biên đủ lớn và việc thêm nó không vi phạm ràng buộc ngân sách.

Trong lượt thứ hai (dòng 16–23), thuật toán tiếp tục cải thiện chất lượng lời giải. Với mỗi phần tử đến e, thuật toán tìm vị trí tối ưu i_e để chèn e (dòng 18). Sau đó, nếu cặp (e, i_e) có lợi ích biên dương, thuật toán sẽ thêm vào nghiệm ứng viên $\mathbf{s}_v$ tương ứng (dòng 19–20). Cụ thể, các bước được trình bày trong Thuật toán 9 và được minh họa trong Hình 3.2.



Hình 3.2: Lưu đồ hoạt động của thuật toán luồng STR.

Trong phần sau, luận án tiến hành phân tích đảm bảo lý thuyết của Thuật toán 9. Luận án phát biểu đảm bảo hiệu năng của thuật toán STR trong định lý sau:

Phác thảo chứng minh. STR loại bỏ giả định biết trước opt bằng cách duy trì đồng thời nhiều nghiệm ứng viên, mỗi nghiệm tương ứng với một giá trị dự đoán v của tối ưu. Ý tưởng chính là giá trị phần tử đơn tốt nhất M tạo ra một khoảng chứa opt, còn lưới hình học $\{(1+\epsilon)^j\}$ bảo đảm tồn tại một giá trị v đủ gần opt. Với giá trị v này, toàn bộ phân tích của STROPT vẫn áp dụng được, chỉ mất thêm một thừa số $(1-\epsilon)$ trong hệ số xấp xỉ. Độ phức tạp truy vấn và bộ nhớ tăng theo số lượng giá trị dự đoán được duy trì.

Algorithm 9 Thuật toán luồng STR

Input: V, f, k, $B_1, B_2, \ldots, B_k$, parameters $\alpha, \epsilon \in (0,1)$.
Output: A solution $\mathbf{s}$ of kSMIK
$M \leftarrow 0$, $(e_{max}, i_{max}) \leftarrow (null, 0)$
for all $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
  $(e_{max}, i_{max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e,i_e),(e_{max},i_{max})\}} f(\mathbf{x})$
  $M \leftarrow f((e_{max}, i_{max}))$
  $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$
  for all $v \in O$ **do**
    $i_v \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v)+c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
    if $\Delta_{(e,i_v)} f(\mathbf{s}_v) \geq \frac{c(e)\alpha v}{B_{i_v}}$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
    end if
  end for
end for
for all $e \in V \setminus supp(\mathbf{s}_v)$ **do**
  for all $v \in O$ **do**
    $i_e \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v)+c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
    if $\Delta_{(e,i)} f(\mathbf{s}_v) > 0$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_e)$
    end if
  end for
end for
$\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \{(e_{max},i_{max})\} \cup \{\mathbf{s}_v : v \in O\}} f(\mathbf{x})$
return $\mathbf{s}$

Định lý 3.2. *Thuật toán 9 thực hiện hai lượt quét qua tập cơ sở, có độ phức tạp truy vấn* $O(\frac{nk}{\epsilon}\log(B))$ *và yêu cầu bộ nhớ* $O(\frac{B}{\epsilon}\log(B))$. *Về hệ số xấp xỉ:*

- Nếu f *là hàm đơn điệu và* $\alpha = \frac{1}{k+1}$, *thì Thuật toán 9 đạt được tỉ lệ xấp xỉ* $\frac{1-\epsilon}{2(k+1)}$.

- Nếu f *là hàm không đơn điệu và* $\alpha = \frac{2}{2k+3}$, *thì Thuật toán 9 đạt được tỉ lệ xấp xỉ* $\frac{1-\epsilon}{2k+3}$.

Chứng minh. Thuật toán thực hiện hai lượt quét. Với mỗi phần tử đến e, thuật toán thực hiện $|O|k$ truy vấn để tìm vị trí tối ưu nhằm quyết định có nên thêm e vào $\mathbf{s}_v, v \in O$ hay không. Do đó, tổng số truy vấn của thuật toán tối đa là:

$$2|O|kn = 2 \cdot \frac{nk\log(B)}{\epsilon} = O\left(\frac{nk\log(B)}{\epsilon}\right).$$

Thuật toán yêu cầu lưu trữ $|O|$ nghiệm ứng viên, mỗi nghiệm yêu cầu B đơn vị bộ nhớ, do đó tổng bộ nhớ là $O(\frac{B}{\epsilon}\log(B))$.

Tiếp theo, ta chứng minh tỉ lệ xấp xỉ của thuật toán. Vì $M\mathsf{opt} \leq BM$, nên tồn tại một giá trị $v = (1+\epsilon)^j$ sao cho:

$$(1-\epsilon)\mathsf{opt} \leq \frac{\mathsf{opt}}{1+\epsilon} \leq v \leq \mathsf{opt}.$$

Do đó, có thể áp dụng Bổ đề 3.2 để thu được hệ số xấp xỉ như đã phát biểu. □

3.1.4. Thực nghiệm

Luận án so sánh các thuật toán đề xuất trên ba ứng dụng sau: **(1)** Tối đa doanh thu sản phẩm theo k-loại dưới ràng buộc chi phí nhóm (kPMIK), **(2**) Bố trí cảm biến theo k-loại dưới ràng buộc chi phí nhóm (kSPIK) và **(3)** Tối đa ảnh hưởng theo k-chủ đề dưới ràng buộc chi phí nhóm (kIMIK). Các thuật toán được so sánh dựa trên ba tiêu chí chính: **(1)** giá trị hàm mục tiêu; **(2)** số lần truy vấn hàm mục tiêu; và **(3)** thời gian thực thi. Như đã được trình bày trong phần nghiên cứu liên quan, hiện tại chưa có công trình nào giải quyết trực tiếp bài toán kSMIK. Do đó, luận án tiến hành so sánh thuật toán STR với phương pháp tham lam truyền thống (Thuật toán 10) để đánh giá hiệu quả.

Ba ứng dụng được lựa chọn nhằm kiểm tra thuật toán trong các bối cảnh khác nhau: kPMIK đại diện cho bài toán trên mạng xã hội có quy mô từ nhỏ đến lớn; kSPIK đại diện cho bài toán cảm biến với số lượng vị trí nhỏ nhưng có ý nghĩa thực tiễn rõ ràng; còn kIMIK đại diện cho bài toán mô phỏng lan truyền ảnh hưởng theo nhiều chủ đề. Cách lựa chọn này giúp đánh giá đồng thời chất lượng nghiệm, chi phí truy vấn và khả năng mở rộng của thuật toán.

Kết quả thực nghiệm cho thấy các thuật toán đề xuất đều đạt hiệu quả cao và ổn định trên nhiều bộ dữ liệu và bối cảnh khác nhau. Đặc biệt, thuật toán STR thường cho nghiệm có chất lượng cạnh tranh trong khi vẫn duy trì mức chi phí truy vấn thấp. Các kết quả đạt được phù hợp với các phân tích lý thuyết đã trình bày, đồng thời cho thấy tính khả thi của các thuật toán khi triển khai trong các ứng dụng thực nghiệm có quy mô và đặc điểm khác nhau.

3.1.4.1. Ứng dụng và bộ dữ liệu

Tối đa doanh thu sản phẩm theo k-loại dưới ràng buộc chi phí nhóm (kPMIK). Dựa vào bài toán tối đa doanh thu *(Định nghĩa 1.26)*, bài toán kPMIK được định nghĩa như sau: Cho đồ thị $G = (V, E)$, trong đó V là tập các đỉnh đại diện cho khách hàng và mỗi cạnh $(i, j) \in E$ có trọng số $w_{ij} \geq 0$ phản ánh mức độ liên hệ giữa hai khách hàng i và j. Mục tiêu là phân bổ k sản phẩm cho khách hàng bằng cách xác định một k-tập $\mathbf{s} = (S_1, S_2, \ldots, S_k)$, trong đó S_i là tập con khách hàng được phân bổ sản phẩm i, sao cho hàm mục tiêu $f(\mathbf{s})$ được cực đại. Hàm mục tiêu được định nghĩa như sau:

$$f(\mathbf{s}) = \sum_{u \in V} \sum_{i=1}^{k} \left(\sum_{v \in S_i} w_{uv} \right)^{\alpha_{u,i}},$$

với $\alpha_{u,i} \in (0, 1)$. Hàm $f(\mathbf{s})$ là một hàm k-submodular và đơn điệu.

Trong thực nghiệm với kPMIK, luận án sử dụng ba bộ dữ liệu: Facebook [83], Astro [81] và Enron [84].

Bố trí cảm biến theo k-loại dưới ràng buộc chi phí nhóm (kSPIK). Dựa vào bài toán Bố trí cảm biến *(Định nghĩa 1.29)*, Bài toán kSPIK được định nghĩa như sau: Cho k loại cảm biến phục vụ các mục đích đo lường khác nhau và một tập V gồm n vị trí, mỗi vị trí chỉ có thể bố trí một cảm biến. Mỗi cảm biến e có chi phí $c(e)$ tùy thuộc vào loại i tương ứng. Với tập các giá trị ngân sách $B_1, B_2, \ldots, B_k$, mục tiêu là bố trí các cảm biến sao cho hàm thông tin thu được là lớn nhất, với điều kiện $c(S_i) = \sum_{e \in S_i} c(e) \leq B_i$ với mọi $i \in [k]$.

Gọi X_e^i là biến ngẫu nhiên đại diện cho thông tin thu thập từ cảm biến loại i tại vị trí e. Với k-tập cảm biến $\mathbf{s}$, thông tin thu được được tính bằng:

$$f(\mathbf{s}) = H\left(\bigcup_{e \in \text{supp}(\mathbf{s})} X_e^i\right),$$

trong đó H là hàm entropy. Hàm f là hàm đơn điệu và k-submodular [98].

Bài toán kSPIK được kiểm chứng thực nghiệm trên tập dữ liệu Intel Lab [12], được thu thập từ 54 cảm biến đặt tại Intel Berkeley Research Lab trong thời gian từ 28/2/2004 đến 5/4/2004.

Tối đa ảnh hưởng theo k-chủ đề dưới ràng buộc chi phí nhóm (kIMIK). Dựa vào bài toán Tối đa ảnh hưởng *(Định nghĩa 1.25)*, Bài toán kIMIK được mô hình hóa trên đồ thị có hướng $G = (V, E)$, trong đó V là tập người dùng và E là các cạnh định hướng. Mỗi cạnh (u, v) có trọng số $w^i(u, v)$ đại diện cho mức độ ảnh hưởng từ u đến v theo chủ đề i. Mỗi người dùng u có ngưỡng ảnh hưởng $\theta^i(u) \in [0, 1]$ đối với mỗi chủ đề.

Mỗi chủ đề i có ngân sách B_i và nhiệm vụ là chọn tập hạt giống S_i sao cho tổng chi phí $c(S_i) \leq B_i$, đồng thời tối đa số người dùng được kích hoạt trong ít nhất một chủ đề. Giá trị mục tiêu là:

$$\max_{\mathbf{s} \in (k+1)^V} \quad \sigma(\mathbf{s}) = \mathbb{E}\left[\left|\bigcup_{i \in [k]} \sigma_i(S_i)\right|\right]$$
$$\text{với điều kiện} \quad c(S_i) \leq B_i, \quad \forall i \in [k].$$

Hàm $\sigma(\cdot)$ là hàm đơn điệu và k-submodular [98].

Thực nghiệm với kIMIK sử dụng bộ dữ liệu ngẫu nhiên ER với $n = 500$, xác suất liên kết $p = 0.1$ và mô hình khuếch tán k-IC [98]. Mỗi giá trị mục tiêu được ước lượng bằng trung bình qua 1.000 lần mô phỏng.

Các cài đặt khác trong thực nghiệm. Trong tất cả các thực nghiệm, luận án thiết lập $\epsilon = 0.1$ và $k = 3$. Không mất tính tổng quát, luận án giả định các ngân sách $B_1, B_2, \ldots, B_k$ là bằng nhau. Tổng ngân sách $B = B_1 + B_2 + \ldots + B_k$ được thiết lập trong khoảng từ 3% đến 30% tổng chi phí của toàn bộ tập đỉnh.

Bảng 3.2: Bảng thống kê các bộ dữ liệu dùng trong thực nghiệm cho bài toán kSMIK

Bộ dữ liệu	**#Đỉnh**	**#Cạnh**	**Phân loại**
Enron [84]	36,692	183,831	Vô hướng
Astro [81]	18,772	198,110	Vô hướng
Facebook [83]	4,039	88,234	Vô hướng
Intel Lab sensors [12]	54	-	-
ER [40]	500	-	Có hướng

Bảng 3.3: Thiết lập thực nghiệm cho bài toán kSMIK

Ứng dụng	**Bộ dữ liệu**	**Thuật toán so sánh**	**Tham số chính**	**Chỉ tiêu đánh giá**
kPMIK	Facebook, Astro, Enron	STR, Greedy	$\epsilon = 0.1$, $k = 3$, B từ 3% đến 30% tổng chi phí	Giá trị hàm mục tiêu, số truy vấn, thời gian chạy
kSPIK	Intel Lab sensors	STR, Greedy	$\epsilon = 0.1$, $k = 3$, ngân sách nhóm bằng nhau	Giá trị hàm thông tin, số truy vấn, thời gian chạy
kIMIK	ER, $n = 500$, $p = 0.1$	STR, Greedy	1.000 lần mô phỏng, mô hình k-IC	Ảnh hưởng kỳ vọng, số truy vấn, thời gian chạy

Algorithm 10 Thuật toán tham lam cho bài toán kSMIK

Input: V, f, k, $B_1, B_2, \ldots, B_k$.
Output: A solution **s** of kSMIK.
s $\leftarrow \emptyset$.
while $V \neq \emptyset$ **do**
  $(e_m, i_m) \leftarrow \arg\max_{e \in V, i \in [k]: c_i(\mathbf{s}) + c(e) \leq B_i} \frac{\Delta_{(e,i)} f(\mathbf{s})}{c(e)}$
  if $(e_m, i_m) = \emptyset$ **then**
    break
  end if
  s $\leftarrow$ **s** $\sqcup (e_m, i_m)$
  $V \leftarrow V \setminus \{e_m\}$
end while
return s

3.1.4.2. Kết quả thực nghiệm

Luận án đánh giá hiệu quả của các thuật toán theo ba chỉ số chính: giá trị hàm mục tiêu, số lượng truy vấn và thời gian thực thi. Hình 3.3, 3.4 và 3.5 trình bày kết quả thực nghiệm đối với bài toán kPMIK trên các bộ dữ liệu Facebook, Astro và Enron.

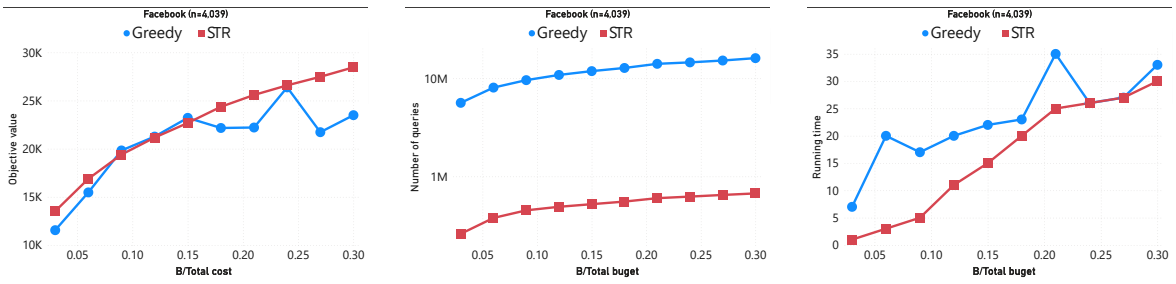
Trong khi đó, Hình 3.6 minh họa kết quả cho bài toán kSPIK trên bộ dữ liệu Intel Lab và Hình 3.7 thể hiện kết quả cho bài toán kIMIK trên bộ dữ liệu Erdos–Renyi (ER).

a) Đối với bài toán kPMIK.

Các hình 3.3(a), 3.4(a) và 3.5(a) trình bày giá trị hàm mục tiêu thu được. Nhìn chung, thuật toán STR liên tục cho kết quả hàm mục tiêu cao hơn so với thuật toán Greedy trên cả ba bộ dữ liệu. Mặc dù có một số trường hợp ngoại lệ mà thuật toán Greedy đạt giá trị mục tiêu cao hơn, nhưng sự chênh lệch này không đáng kể (với $B = 0.09$ và 0.15 trên tập Facebook; $B = 0.06$ và 0.18 trên tập Astro). Đáng chú ý, thuật toán STR thể hiện xu hướng tăng đều của giá trị mục tiêu khi giá trị B tăng. Ngược lại, thuật toán Greedy có sự gia tăng nhanh ban đầu nhưng dần chững lại khi B tiếp tục tăng. Đặc biệt, trên bộ dữ liệu Enron, giá trị mục tiêu của STR cao hơn trung bình 16% so với Greedy khi $B \geq 0.15$.

Các hình 3.3(b), 3.4(b) và 3.5(b) minh họa số lượng truy vấn hàm $f(\cdot)$. Thuật toán STR luôn yêu cầu số lượng truy vấn ít hơn đáng kể so với Greedy trên cả ba bộ dữ liệu. Cụ thể, trên tập Facebook, Greedy thực hiện số truy vấn gấp từ 21.2 đến 23.9 lần; trên tập Astro từ 92.4 đến 103.6 lần; và trên tập Enron từ 63.5 đến 130.8 lần so với STR. Đáng lưu ý, số lượng truy vấn của Greedy tăng nhanh khi B tăng, trong khi STR duy trì số lượng ổn định hơn.

Thời gian thực thi được thể hiện trong các hình 3.3(c), 3.4(c) và 3.5(c). Thuật toán Greedy luôn có thời gian thực thi lớn hơn đáng kể so với STR. Cụ thể, trên tập Facebook, Greedy chậm hơn từ 1.1 đến 7.0 lần; trên tập Astro từ 1.1 đến 1.5 lần; và trên tập Enron từ 31.1 đến 47.3 lần. Sự khác biệt này đặc biệt rõ rệt trên tập Enron do số lượng đỉnh lớn, qua đó cho thấy lợi thế của STR về thời gian chạy trong thiết lập thực nghiệm này.

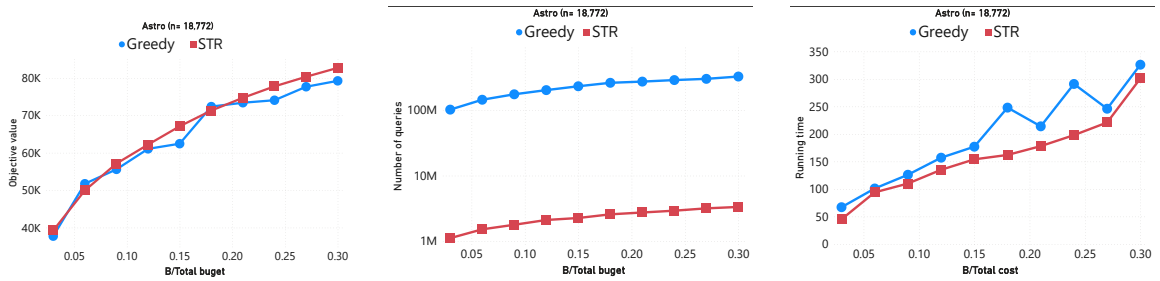


(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.3: Kết quả thực nghiệm trên tập dữ liệu Facebook cho ứng dụng kPMIK.

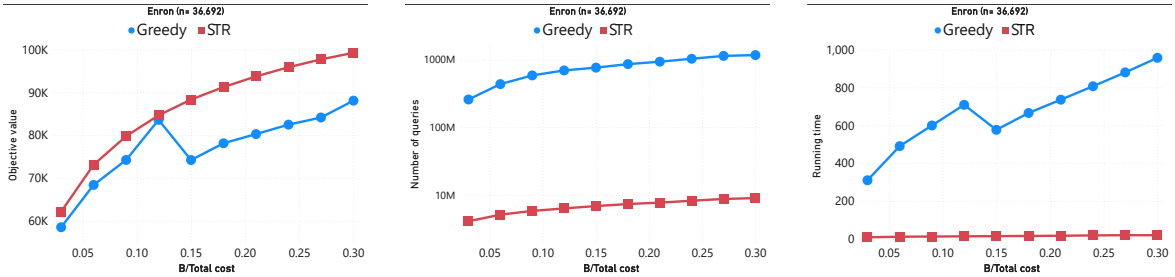
b) Đối với bài toán kSPIK.

Hình 3.6(a) trình bày giá trị hàm mục tiêu. Kết quả cho thấy giá trị mục tiêu giữa hai thuật toán STR và Greedy là tương đương nhau, mặc dù thuật toán Greedy luôn đạt giá trị cao hơn một chút. Hình 3.6(b) minh họa số lượng truy vấn. Thuật toán Greedy



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

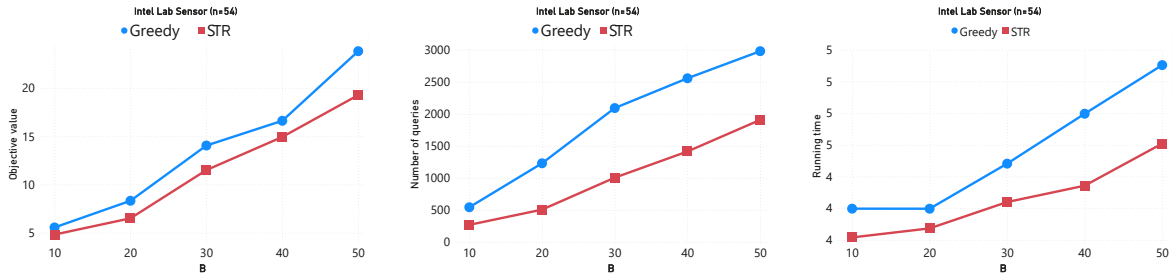
Hình 3.4: Kết quả thực nghiệm trên tập dữ liệu Astro cho ứng dụng kPMIK.



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.5: Kết quả thực nghiệm trên tập dữ liệu Enron cho ứng dụng kPMIK.

liên tục yêu cầu nhiều truy vấn hơn so với STR ở mọi mức ngân sách. Cụ thể, tại các mức $B = 10$ và $B = 20$, Greedy cần số truy vấn nhiều hơn khoảng 1.5 lần. Khi $B \geq 30$, số truy vấn của Greedy luôn gấp đôi so với STR. Hình 3.6(c) trình bày thời gian thực thi. Do bộ dữ liệu Intel Lab chỉ gồm 54 đỉnh nên thời gian chạy của hai thuật toán gần như tương đương. Tuy nhiên, thời gian thực thi của STR vẫn thấp hơn một cách ổn định so với Greedy.



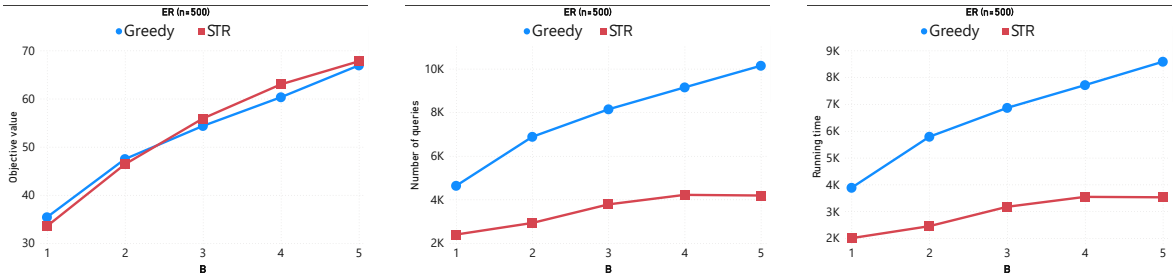
(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.6: Kết quả thực nghiệm trên tập dữ liệu Intel Lab cho ứng dụng kSPIK.

c) Đối với bài toán kIMIK.

Tương tự hai ứng dụng trước, ba chỉ số gồm giá trị hàm mục tiêu, số lượng truy vấn và thời gian thực thi đều thể hiện xu hướng nhất quán. Cụ thể, giá trị hàm mục tiêu

của hai thuật toán gần như trùng khớp, chỉ dao động nhẹ tại một số điểm và sự khác biệt này có thể xem là không đáng kể. Về số lượng truy vấn, như thể hiện trong Hình 3.7(b), thuật toán Greedy luôn yêu cầu nhiều truy vấn hơn đáng kể, cao gấp từ hai đến ba lần so với STR. Đặc biệt, số lượng truy vấn của Greedy tăng nhanh khi ngân sách B tăng, trong khi số truy vấn của STR chỉ tăng nhẹ. Đối với thời gian thực thi (Hình 3.7(c)), xu hướng này phản ánh tương quan trực tiếp với số lượng truy vấn, cho thấy mối quan hệ giữa thời gian chạy và chi phí truy vấn trong quá trình thực thi thuật toán.



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.7: Kết quả thực nghiệm trên tập dữ liệu ER cho ứng dụng kIMIK.

Bảng 3.4: Tổng hợp nhận xét chính từ thực nghiệm cho bài toán kSMIK

Ứng dụng	**Giá trị hàm mục tiêu**	**Số truy vấn**	**Nhận xét**
kPMIK	STR thường cao hơn Greedy, đặc biệt rõ trên Enron khi ngân sách tăng.	STR yêu cầu ít truy vấn hơn rõ rệt trên cả ba bộ dữ liệu.	Phù hợp với bối cảnh mạng lớn, nơi chi phí truy vấn là yếu tố quan trọng.
kSPIK	Hai thuật toán cho giá trị gần tương đương; Greedy nhỉnh hơn nhẹ ở một số điểm.	STR cần ít truy vấn hơn khi ngân sách tăng.	Dữ liệu nhỏ nên khác biệt thời gian chạy không lớn.
kIMIK	Giá trị mục tiêu gần tương đương giữa hai thuật toán.	Greedy cần nhiều truy vấn hơn, thường cao hơn từ hai đến ba lần.	STR thể hiện lợi thế về chi phí tính toán trong mô phỏng lan truyền.

Tổng hợp các kết quả trong Bảng 3.4 cho thấy ưu điểm chính của STR nằm ở khả năng kiểm soát số lượng truy vấn và thời gian chạy khi dữ liệu lớn hoặc ngân sách tăng. Về chất lượng nghiệm, thuật toán thường đạt giá trị cạnh tranh với Greedy và trong một số thiết lập cho giá trị cao hơn, nhưng không phải mọi bộ dữ liệu đều cho cùng một mức cải thiện. Điều này phản ánh đúng đặc điểm của thuật toán luồng: ưu thế nổi bật là khả năng mở rộng và chi phí truy vấn thấp, trong khi chất lượng nghiệm phụ thuộc vào cấu trúc dữ liệu và mức ngân sách cụ thể.

Phần trên đã trình bày bài toán trong đó mỗi phần tử được gán vào nhiều nhất một nhóm và mỗi nhóm có ngân sách riêng. Phần tiếp theo xét một kiểu mở rộng khác: một

phần tử có thể được chọn nhiều lần với mức độ nguyên khác nhau, còn tổng mức lựa chọn bị giới hạn bởi một ràng buộc lực lượng. Cách chuyển từ ràng buộc theo nhóm sang ràng buộc theo mức độ lựa chọn giúp hoàn thiện mạch nghiên cứu của chương về các bài toán tối ưu submodular mở rộng dưới ràng buộc tài nguyên có cấu trúc.

3.2. Bài toán tối đa hàm DR-submodular với ràng buộc lực lượng

3.2.1. Định nghĩa bài toán

Hàm DR-submodular *(Định nghĩa 1.16)* là một mở rộng tự nhiên của hàm submodular truyền thống sang không gian lưới nguyên $\mathbb{Z}_+^V$. Trong chương 2 và 4, luận án đã xem xét bài toán tối ưu hàm submodular, trong đó mỗi phần tử chỉ được chọn một lần. Tuy nhiên, trong nhiều ứng dụng thực tiễn, việc cho phép lựa chọn lặp lại một phần tử với cường độ khác nhau là rất quan trọng. Những bối cảnh như phân bổ ngân sách, hoạch định sản xuất, hoặc lan truyền ảnh hưởng đa chủ đề đòi hỏi mô hình hóa tối ưu hóa với biến đa trị. Do đó, việc mở rộng bài toán sang dạng tối đa hàm DR-submodular với ràng buộc lực lượng là một hướng phát triển tự nhiên và cần thiết để đáp ứng các yêu cầu thực tế phức tạp.

Bài toán tối đa DR-submodular với ràng buộc lực lượng (ký hiệu là DrSMC) được phát biểu như sau:

Định nghĩa 3.2 (Bài toán tối đa DR-submodular với ràng buộc lực lượng – DrSMC)**.** Cho một hàm DR-submodular $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, một lưới nguyên bị chặn bởi vectơ $\mathbb{B} \in \mathbb{Z}_+^V$ và một số nguyên dương $k > 0$. Bài toán yêu cầu tìm vectơ $\mathbf{x} \leq \mathbb{B}$ sao cho tổng kích thước $\|\mathbf{x}\|_1 \leq k$ và giá trị $f(\mathbf{x})$ đạt cực đại, tức là:

$$\max_{\mathbf{x} \in \mathbb{Z}_+^V} \quad f(\mathbf{x}) \quad \text{s.t.} \quad \|\mathbf{x}\|_1 \leq k, \quad \mathbf{0} \leq \mathbf{x} \leq \mathbb{B}$$

với $\mathbb{B} = k \cdot \mathbf{1}$ và $\|\mathbf{x}\|_1 = \sum_{e \in V} \mathbf{x}(e)$.

Một ví dụ cụ thể cho bài toán DrSMC là bài toán tối đa doanh thu trong chiến lược phân phối sản phẩm. Giả sử một công ty có thể phân phối nhiều sản phẩm với số lượng khác nhau đến các khu vực thị trường, trong khi tổng số lượng phân phối bị giới hạn bởi một ngân sách k. Mỗi sản phẩm mang lại lợi nhuận khác nhau tùy theo số lượng phân phối, tuy nhiên, lợi ích biên thường giảm dần khi số lượng tăng lên – phản ánh hiệu ứng bão hòa thị trường. Bài toán yêu cầu xác định số lượng từng sản phẩm cần phân phối sao cho tổng lợi nhuận là lớn nhất, đồng thời đảm bảo không vượt quá ngân sách cho phép. Đây là một mô hình điển hình của tối ưu DR-submodular với ràng buộc lực lượng.

Lưới nguyên có thể được biểu diễn như một đa tập có kích thước $O(nk)$, cho phép việc sử dụng trực tiếp các thuật toán hiện có cho bài toán Tối đa Hàm submodular với ràng buộc lực lượng (SMC). Tuy nhiên, thuật toán nhanh nhất được biết đến cho SMC, được đề xuất bởi Buchbinder và cộng sự [15], yêu cầu $\Omega(nk)$ truy vấn hàm f, điều này

không phải là đa thức theo kích thước đầu vào. Đặc biệt, vì k là một số nguyên và có thể được mã hóa chỉ bằng $O(\log k)$ bit, nên bất kỳ thuật toán nào có độ phức tạp $O(\text{poly}(k))$ cũng không đảm bảo là đa thức theo kích thước đầu vào.

Gần đây, Ene và Nguyen [34] đã đề xuất một kỹ thuật ánh xạ để chuyển bài toán tối ưu DR-submodular thành bài toán tối ưu submodular tiêu chuẩn, từ đó cho phép áp dụng các kết quả hiện có. Cụ thể, sau khi áp dụng phép ánh xạ, bài toán DrSMC được chuyển thành bài toán SMK *(Chương 2)*:

$$\max_{S \subseteq V'} g(S) \quad \text{với điều kiện:} c(S) \leq k,$$

trong đó $V' = \bigcup_{e \in E} \{(e, j) : j \in [t_e]\}$ là tập cơ sở mở rộng được tạo bằng cách phân rã mỗi biến nguyên $\mathbf{x}(e)$ thành tổng các trọng số nhỏ hơn $a_{e,j} \leq \epsilon k$. Hàm submodular $g(S)$ được định nghĩa là $g(S) := f(\mathbf{x})$ với $\mathbf{x}(e) = \sum_{j:(e,j) \in S} a_{e,j}$ và hàm chi phí là $c(S) := \sum_{(e,j) \in S} a_{e,j}$. Do phép ánh xạ này, kích thước không gian tìm kiếm mới trở thành $O(n \log k)$ thay vì $O(n)$ như ban đầu. Đối với bài toán SMK, như đã trình bày trong Chương 2 của luận án, trong số các thuật toán có độ phức tạp truy vấn tuyến tính, hai thuật toán ngẫu nhiên RLA (Thuật toán 1) và SMKRANACC [53] đạt tỉ lệ xấp xỉ tốt nhất là $\frac{1}{4} - \epsilon$. Tuy tỉ lệ xấp xỉ được bảo toàn sau ánh xạ, nhưng độ phức tạp truy vấn tăng đáng kể do mở rộng tập cơ sở (xem Bảng 3.5). Điều này làm suy giảm tính mở rộng thực tế của các thuật toán, đặc biệt khi k lớn hoặc ϵ nhỏ. Đặc biệt, một điểm yếu của các thuật toán ngẫu nhiên là chúng chỉ đảm bảo tỉ lệ xấp xỉ theo kỳ vọng và kết quả thực nghiệm trên tập dữ liệu thực tế có thể không ổn định.

Ngược lại, các nghiên cứu gần đây cho thấy các thuật toán xác định thường có hiệu suất thực nghiệm ổn định trên nhiều tập dữ liệu, nhờ đó trở thành một hướng tiếp cận đáng quan tâm [22, 52, 74, 86]. Hơn nữa, khi dữ liệu ngày càng mở rộng với quy mô lớn, việc thiết kế các thuật toán xấp xỉ cho DrSMC sao cho giảm thiểu số lần truy vấn oracle là ngày càng quan trọng để đảm bảo hiệu quả tính toán.

Theo hiểu biết của nghiên cứu sinh, đến thời điểm hiện tại chưa có công trình nào giải quyết trực tiếp bài toán tối đa hàm DR-submodular với ràng buộc lực lượng. Từ thực tế đó, hai câu hỏi nghiên cứu quan trọng được đặt ra: (1) Liệu có thể thiết kế các thuật toán mới đạt tỉ lệ xấp xỉ tương đương nhưng cải thiện đáng kể độ phức tạp truy vấn? (2) Liệu có thể xây dựng một thuật toán tất định cho bài toán DR-submodular với ràng buộc lực lượng có đảm bảo xấp xỉ tương đương với các phương pháp ngẫu nhiên hiện có? Các thuật toán được đề xuất trong phần tiếp theo sẽ nhằm giải quyết hai câu hỏi này.

3.2.2. Nghiên cứu liên quan

Trong phần này, luận án trình bày tổng quan các nghiên cứu hiện có liên quan đến thuật toán tối ưu hàm DR-submodular. Các công trình trước đây có thể được phân loại

tổng quát dựa trên tính chất đơn điệu của hàm mục tiêu, vì hai thiết lập này đặt ra các thách thức thuật toán khác nhau về bản chất.

Hàm DR-submodular đơn điệu. Soma và Yoshida [112] là những người đầu tiên mở rộng khái niệm hàm submodular lên lưới nguyên bằng cách giới thiệu khái niệm DR-submodularity và đề xuất một thuật toán xấp xỉ hai tiêu chí, đồng thời xét đến cả hàm mục tiêu và hàm chi phí. Sau đó, trong một công trình tiếp theo, Soma và cộng sự [113] đã phát triển hai thuật toán xác định đạt được tỉ lệ xấp xỉ $1 - \frac{1}{e} - \epsilon$ cho bài toán tối đa các hàm DR-submodular đơn điệu và các hàm submodular trên lưới tổng quát, với độ phức tạp thời gian lần lượt là $O\left(\frac{n}{\epsilon} \log k \log \frac{k}{\epsilon}\right)$ và $O\left(\frac{n}{\epsilon^2} \log k \log \frac{k}{\epsilon} \log \tau\right)$, trong đó k là ngân sách kích thước và giới hạn tối đa của từng tọa độ và τ là tỉ số giữa giá trị hàm lớn nhất và lợi ích biên dương nhỏ nhất. Lai và các cộng sự (2019) [79] đã giới thiệu một thuật toán ngẫu nhiên, tuy nhiên thuật toán này gặp vấn đề về tính không ổn định trong số lượng truy vấn hàm f. Gần đây hơn, Schiabel và các cộng sự (2025) [109] đã đề xuất thuật toán Stochastic Greedy Lattice (SGL), mở rộng chiến lược chọn ngẫu nhiên tham lam cho các hàm DR-submodular đơn điệu trên lưới nguyên. Thuật toán SGL đạt được đảm bảo xấp xỉ $1 - \frac{1}{e} - \hat{t}\epsilon$ với xác suất lớn hơn $\frac{1}{2}$, trong đó $\hat{t}$ là một hằng số nhỏ phụ thuộc vào số vòng lặp.

Hàm DR-submodular không đơn điệu. Thiết kế các thuật toán cho hàm DR-submodular không đơn điệu là một thách thức lớn hơn đáng kể so với trường hợp đơn điệu do khả năng giảm giá trị hàm. Gottschalk và Peis (2015) [48] đã đề xuất thuật toán tham lam kép để tối đa các hàm submodular trên lưới nguyên bị chặn, đạt được tỉ lệ xấp xỉ $\frac{1}{3}$, mặc dù có độ phức tạp giả đa thức. Ene và cộng sự (2020) [33] đã phát triển một thuật toán song song cho bài toán tối đa DR-submodular không đơn điệu, nhưng bị giới hạn trong miền liên tục. Li và cộng sự (2023) [87] tiếp tục cải thiện hiệu quả thuật toán bằng cách sử dụng phương pháp song song.

Như đã đề cập trong phần mở đầu, bằng cách áp dụng kỹ thuật rút gọn được đề xuất bởi Ene và Nguyen [34], bài toán DrSMC có thể được chuyển đổi thành một bài toán tối đa submodular với ràng buộc chi phí (SMK). Trong phần tiếp theo, luận án nhắc lại hướng nghiên cứu tập trung vào bài toán SMK. Công trình đầu tiên giải quyết bài toán SMK trong trường hợp không đơn điệu được đề xuất bởi Lee và cộng sự [80], đạt được tỉ lệ xấp xỉ $\frac{1}{5} - \epsilon$ với độ phức tạp truy vấn đa thức. Sau đó, nhiều nghiên cứu tiếp theo đã hướng đến việc cải thiện hiệu quả thuật toán [14, 50, 53, 85, 90, 114]. Trong số đó, phương pháp của Buchbinder và cộng sự [14] đạt tỉ lệ xấp xỉ tốt nhất là $\frac{1}{2.6}$, mặc dù chi phí truy vấn oracle rất cao. Ngược lại, thuật toán RLA (trong Chương 2) đạt độ phức tạp truy vấn thấp nhất – tuyến tính theo kích thước đầu vào – trong khi vẫn đạt được tỉ lệ xấp xỉ $\frac{1}{4} - \epsilon$.

Bảng 3.5: Bảng so sánh các thuật toán cho bài toán DrSMC.

Thuật toán	**Tỉ lệ xấp xỉ**	**Độ phức tạp**	**Phân loại**
RLA+Reduction [34]	$\frac{1}{4} - \epsilon$	$O\left(\frac{n}{\epsilon}\log(k)\log(\frac{1}{\epsilon})\right)$	Ngẫu nhiên
SMKRANACC [53]+Reduction [34]	$\frac{1}{4} - \epsilon$	$O\left(\frac{n}{\epsilon}\log(k)\log(\frac{k}{\epsilon})\right)$	Ngẫu nhiên
FastDrSub	0.044	$O(n\log k)$	Tất định
FastDrSub+	$\frac{1}{4} - \epsilon$	$O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(k))$	Tất định

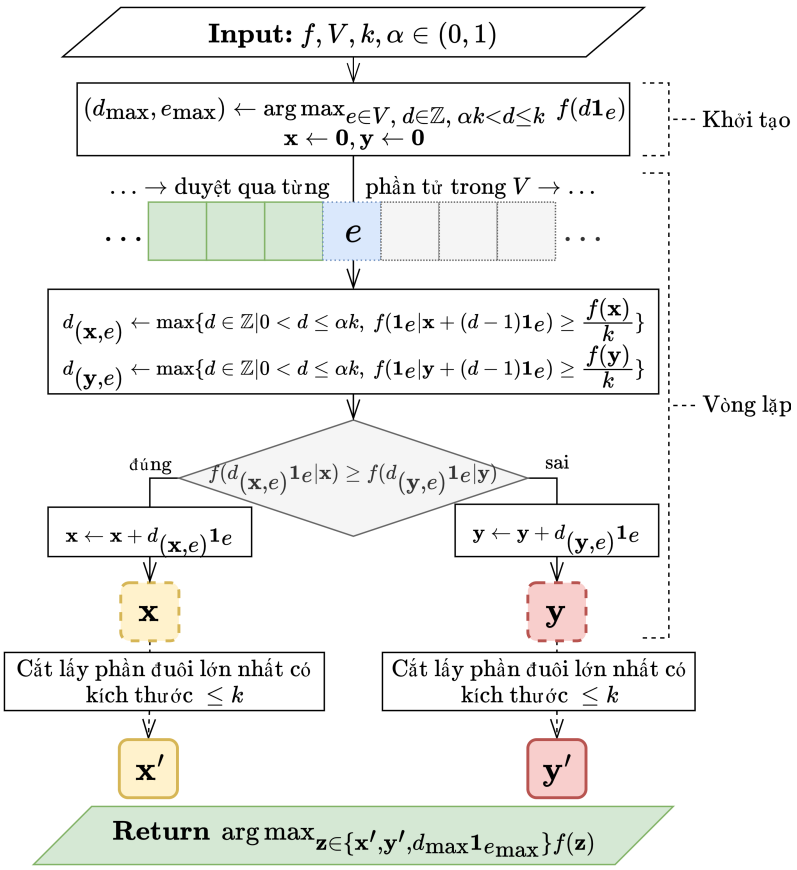
Mỗi thuật toán được đánh giá dựa trên bảo đảm xấp xỉ và số lượng truy vấn cần thiết. Trong đó $\epsilon > 0$ là tham số độ chính xác, k là ràng buộc lực lượng và $\|\mathbf{b}\|_\infty = k$ là giới hạn tọa độ lớn nhất. Đặc biệt, độ phức tạp của RLA và SMKRANACC được xác định lại bằng cách sử dụng khung giảm của Ene và Nguyen [34], trong đó kích thước tập cơ sở được tăng lên thành $n(2\log k + 1)$.

3.2.3. Thuật toán đề xuất

Trong phần này, luận án trình bày hai thuật toán hiệu quả nhằm giải bài toán DrSMC: thuật toán nhanh cho bài toán tối đa hàm DR-submodular với ràng buộc lực lượng (FastDrSub) và thuật toán nhanh hiệu quả cho bài toán tối đa hàm DR-submodular với ràng buộc lực lượng (FastDrSub+). Thuật toán FastDrSub đạt được tỉ lệ xấp xỉ là 0.044 (được chứng minh trong Định lý 3.3) với độ phức tạp truy vấn là $O(n\log k)$, với n là kích thước của tập cơ sở và k là ràng buộc về kích thước. Dựa trên FastDrSub, thuật toán FastDrSub+ nâng cao hiệu quả tối ưu thông qua việc sử dụng số lượng hữu hạn giả định cho giá trị tối ưu cùng với phương pháp ngưỡng tham lam [6]. Nhờ đó, FastDrSub+ cải thiện đáng kể tỉ lệ xấp xỉ lên gần $\frac{1}{4}$ (theo Định lý 3.4) mà không làm tăng độ phức tạp truy vấn, vẫn giữ ở mức $O(n\log k)$. Hai thuật toán này đều được thiết kế nhằm đạt được sự cân bằng giữa chất lượng nghiệm và hiệu suất tính toán, đặc biệt trong bối cảnh các bài toán submodular phi đơn điệu với ràng buộc lực lượng ngày càng phổ biến trong các ứng dụng học máy và tối ưu tổ hợp hiện đại.

3.2.3.1. Thuật toán FastDrSub

Thuật toán FastDrSub(Thuật toán 11) nhận vào một hàm DR-submodular f, một tập cơ sở hữu hạn V có kích thước n, một ngân sách nguyên dương k và một tham số ngưỡng $\alpha \in (0, 1)$. Mục tiêu của thuật toán là tìm một vector nghiệm $\mathbf{z} \in \mathbb{Z}_+^V$ sao cho $\|\mathbf{z}\|_1 \leq k$, nhằm xấp xỉ tối đa giá trị của $f(\mathbf{z})$. Ý tưởng cốt lõi của thuật toán là đồng thời xây dựng hai vector nghiệm rời nhau, ký hiệu $\mathbf{x}$ và $\mathbf{y}$, theo một quy tắc ngưỡng động, từ đó cho phép phân tích lý thuyết rõ ràng dựa trên Bổ đề 1.1.



Hình 3.8: Lưu đồ hoạt động của thuật toán FastDrSub

Cụ thể, thuật toán khởi đầu bằng việc thực hiện tìm kiếm nhị phân trên miền $d \in (\alpha k, k]$ để xác định cặp $(d_{\max}, e_{\max})$ sao cho $f(d\mathbf{1}_e)$ là lớn nhất; lựa chọn này được giữ lại như một ứng viên đơn lẻ độc lập (Dòng 1). Tiếp theo, với khởi tạo $\mathbf{x} = \mathbf{y} = \mathbf{0}$, thuật toán lần lượt duyệt qua từng phần tử $e \in V$. Với mỗi phần tử, thuật toán xác định số nguyên lớn nhất $d \leq \alpha k$ sao cho độ lợi biên thoả mãn $f(\mathbf{1}_e \mid \mathbf{x} + (d-1)\mathbf{1}_e) \geq \frac{f(\mathbf{x})}{k}$ (tương tự đối với $\mathbf{y}$) và chỉ cập nhật vector có độ lợi biên lớn hơn.

Nhờ đó, hai vector $\mathbf{x}$ và $\mathbf{y}$ phát triển theo các quỹ đạo rời nhau, được điều hướng bởi cùng một ngưỡng phụ thuộc giá trị. Sau khi xử lý toàn bộ phần tử, thuật toán cắt bỏ các đơn vị được thêm gần nhất từ mỗi vector để thu được $\mathbf{x}'$ và $\mathbf{y}'$, sao cho $\|\mathbf{x}'\|_1 \leq k$ và $\|\mathbf{y}'\|_1 \leq k$. Cuối cùng, thuật toán trả về nghiệm tốt nhất trong số ba lựa chọn: $\mathbf{x}'$, $\mathbf{y}'$ và $d_{\max}\mathbf{1}_{e_{\max}}$. Cụ thể, các bước được trình bày trong Thuật toán 11 và được minh họa trong Hình 3.8.

Để phân tích bảo đảm lý thuyết của thuật toán FastDrSub, trước tiên luận án định nghĩa hoặc nhắc lại các ký hiệu cần thiết trong Bảng 3.6.

Algorithm 11 Thuật toán FastDrSub

Input: $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$,$V$, k, $\alpha \in (0, 1)$
Output: A solution **s** of DrSMC

1 $(d_{max}, e_{max}) \leftarrow \arg\max_{e \in V,\ d \in \mathbb{Z},\ \alpha k < d \leq k} f(d\mathbf{1}_e)$ by using the binary search
2 $\mathbf{x} \leftarrow \mathbf{0}$, $\mathbf{y} \leftarrow \mathbf{0}$
3 **foreach** $e \in V$ **do**
4 $\quad d_{(\mathbf{x},e)} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq \alpha k,\ f(\mathbf{1}_e \mid \mathbf{x} + (d-1)\mathbf{1}_e) \geq \frac{f(\mathbf{x})}{k}\}$ by using the binary search
5 $\quad d_{(\mathbf{y},e)} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq \alpha k,\ f(\mathbf{1}_e \mid \mathbf{y} + (d-1)\mathbf{1}_e) \geq \frac{f(\mathbf{y})}{k}\}$ by using the binary search
6 $\quad$ **if** $f(d_{(\boldsymbol{x},e)}\mathbf{1}_e \mid \boldsymbol{x}) \geq f(d_{(\boldsymbol{y},e)}\mathbf{1}_e \mid \boldsymbol{y})$ **then**
7 $\quad\quad \mathbf{x} \leftarrow \mathbf{x} + d_{(\mathbf{x},e)}\mathbf{1}_e$
8 $\quad$ **else**
9 $\quad\quad \mathbf{y} \leftarrow \mathbf{y} + d_{(\mathbf{y},e)}\mathbf{1}_e$
10 Define $\{e_1, e_2, \ldots, e_t\}$ as the set of last t elements added into **x**, i.e., $\mathbf{x}_t = \sum_{i=1}^{t} \mathbf{x}(e_t)\mathbf{1}_{e_t}$
11 $\mathbf{x}' \leftarrow \arg\max_{\mathbf{x}_t : |\mathbf{x}_t| \leq k} t$
12 Define $\{e'_1, e'_2, \ldots, e'_t\}$ as the set of last t elements added into **y**, i.e., $\mathbf{y}_t = \sum_{i=1}^{t} \mathbf{y}(e'_t)\mathbf{1}_{e'_t}$
13 $\mathbf{y}' \leftarrow \arg\max_{\mathbf{y}_t : |\mathbf{y}_t| \leq k} t$
14 $\mathbf{s} \leftarrow \arg\max_{\mathbf{z} \in \{\mathbf{x}', \mathbf{y}', d_{max}\mathbf{1}_{e_{max}}\}} f(\mathbf{z})$
15 **return s**

Bảng 3.6: Bảng các ký tự toán học dùng trong phân tích thuật toán FastDrSub

Ký hiệu	Ý nghĩa
$\mathbf{o}$	$\mathbf{o}$ là vector nghiệm tối ưu và $\mathsf{opt} = f(\mathbf{o})$
$\mathbf{o}_1$	Là một vector con của $\mathbf{o}$, sao cho $\mathbf{o}_1(e) = \mathbf{o}(e)$ nếu $\mathbf{o}(e) \leq \alpha k$ và $\mathbf{o}_1(e) = 0$ nếu ngược lại với mọi $e \in V$
$\mathbf{o}_2$	Là một vector con khác của $\mathbf{o}$, sao cho $\mathbf{o}_2(e) = \mathbf{o}(e)$ nếu $\mathbf{o}(e) > \alpha k$ và $\mathbf{o}_1(e) = 0$ nếu ngược lại với mọi $e \in V$
$V_{\mathbf{x}}$	$V_{\mathbf{x}} = \{e \in V : \mathbf{x}(e) > 0, \mathbf{o}(e) = 0\}$, $V_{\mathbf{x} \wedge \mathbf{o}} = \{e \in V : \mathbf{x}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{x} \wedge \mathbf{o}}$	$V_{\mathbf{x} \wedge \mathbf{o}} = \{e \in V : \mathbf{x}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{y}}$	$V_{\mathbf{y}} = \{e \in V : \mathbf{y}(e) > 0, \mathbf{o}(e) = 0\}$, $V_{\mathbf{y} \wedge \mathbf{o}} = \{e \in V : \mathbf{y}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{y} \wedge \mathbf{o}}$	$V_{\mathbf{y} \wedge \mathbf{o}} = \{e \in V : \mathbf{y}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{o}}$	$V_{\mathbf{o}} = \{e \in V : \mathbf{o}(e) > 0, \mathbf{o}(e) > \mathbf{x}(e), \mathbf{o}(e) > \mathbf{y}(e)\}$
t_e	Với $e \in V_{\mathbf{x}} \cup V_{\mathbf{y}}$, ký hiệu $t_e = \mathbf{o}_1(e) - \mathbf{z}(e)$, trong đó $\mathbf{z} \in \{\mathbf{x}, \mathbf{y}\}$ (lưu ý $\mathbf{x} \wedge \mathbf{y} = \mathbf{0}$)
$\mathbf{x}_e$	$\mathbf{x}_e$ Giá trị của **x** ngay trước khi e được thêm vào **x** tại dòng 7
$\mathbf{y}_e$	$\mathbf{y}_e$ Giá trị của **y** ngay trước khi e được thêm vào **y** tại dòng 9

Phác thảo chứng minh. Phân tích của FastDrSub chia nghiệm tối ưu **o** thành hai phần theo số lần chọn của từng phần tử: phần $\mathbf{o}_1$ gồm các tọa độ không vượt quá ngưỡng αk và phần $\mathbf{o}_2$ gồm các tọa độ lớn. Với $\mathbf{o}_1$, thuật toán xây dựng hai vectơ **x** và **y** theo quy tắc tham lam, từ đó chứng minh rằng giá trị của $\mathbf{o}_1$ khi ghép với **x**, **y** bị chặn bởi tổng giá trị của hai vectơ này. Với $\mathbf{o}_2$, vì mỗi tọa độ lớn chiếm một phần đáng kể ngân sách lực

lượng, số lượng phần tử như vậy bị giới hạn và có thể được chặn thông qua bước chọn phần tử đơn lớn. Kết hợp hai cận này cho ra hệ số xấp xỉ của FastDrSub.

Từ các định nghĩa trong Bảng 3.6, ta thiết lập mối quan hệ giữa $\mathbf{x}$, $\mathbf{y}$ và $\mathbf{o}_1$ như sau:

Bổ đề 3.4. *Ta có:* $f(\boldsymbol{o}_1 \vee \boldsymbol{x}) + f(\boldsymbol{o}_1 \vee \boldsymbol{y}) \leq 4(f(\boldsymbol{x}) + f(\boldsymbol{y}))$.

Chứng minh. Xét các phần tử $e \in V_{\mathbf{o}}$, theo quy tắc chọn $d_{(\mathbf{x},e)}$ (dòng 4, 5) trong mỗi vòng lặp, ta có $f(\mathbf{1}_e \mid \mathbf{x}_e + \mathbf{x}(e)\mathbf{1}_e) < \frac{f(\mathbf{x}_e)}{k}$. Do đó,

$$\begin{aligned} f(t_e\mathbf{1}_e \mid \mathbf{x}_e + \mathbf{x}(e)\mathbf{1}_e) &\leq t_e f(\mathbf{1}_e|\mathbf{x}_e + \mathbf{x}(e)\mathbf{1}_e) \quad \text{(Theo Bổ đề 1.2)} \\ &\leq \frac{t_e f(\mathbf{x}_e)}{k} \leq \frac{\mathbf{o}_1(e) f(\mathbf{x}_e)}{k}. \end{aligned}$$

Mỗi phần tử $e \in V_{\mathbf{y}\wedge\mathbf{o}}$ thỏa mãn quy tắc chọn trong dòng 6-10. Ta xét hai trường hợp. Nếu $d_{(\mathbf{x},e)} \geq d_{(\mathbf{y},e)}$, ta có:

$$f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{x}) \leq f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{x}) + \sum_{i=1}^{d_{(\mathbf{x},e)}-d_{(\mathbf{y},e)}} f(\mathbf{1}_e|\mathbf{x} + d_{(\mathbf{y},e)}\mathbf{1}_e + (i-1)\mathbf{1}_e) \tag{3.5}$$

$$\leq f(d_{(\mathbf{x},e)}\mathbf{1}_e|\mathbf{x}) \tag{3.6}$$

$$\leq f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{y}) \tag{3.7}$$

trong đó bất đẳng thức (3.5) là do quy tắc chọn $d_{(\mathbf{x},e)}$, tức là $f(\mathbf{1}_e \mid \mathbf{x} + (d-1)\mathbf{1}_e) \geq \frac{f(\mathbf{x})}{k} \geq 0$; bất đẳng thức (3.5) cũng do quy tắc chọn phần tử e.

Nếu $d_{(\mathbf{x},e)} < d_{(\mathbf{y},e)}$, đặt $l_e = d_{(\mathbf{y},e)} - d_{(\mathbf{x},e)}$ thì ta có:

$$\begin{aligned} f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{x}) &= f(d_{(\mathbf{x},e)}\mathbf{1}_e|\mathbf{x}) + f(l_e\mathbf{1}_e|\mathbf{x} + d_{(\mathbf{x},e)}\mathbf{1}_e) \\ &\leq f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{y}) + \frac{\mathbf{y}(e) f(\mathbf{x})}{k}. \end{aligned}$$

Tổng hợp lại, ta có:

$$f(\mathbf{o}_1 \vee \mathbf{x}) - f(\mathbf{x}) \leq \sum_{e\in V_{\mathbf{o}_1}} f(t_e\mathbf{1}_e|\mathbf{x}) + \sum_{e\in V_{\mathbf{y}\wedge\mathbf{o}_1}} f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{x}) \tag{3.8}$$

$$\leq \sum_{e\in V_{\mathbf{o}_1}} \frac{\mathbf{o}_1(e) f(\mathbf{x}_e)}{k} + \sum_{e\in V_{\mathbf{y}\wedge\mathbf{o}_1}} \left(f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{y}) + \frac{\mathbf{y}(e) f(\mathbf{x})}{k} \right) \tag{3.9}$$

$$\leq f(\mathbf{x}) + \sum_{e\in V_{\mathbf{y}\wedge\mathbf{o}_1}} \left(f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{y}_e) + \frac{\mathbf{y}(e) f(\mathbf{x})}{k} \right) \tag{3.10}$$

$$\leq 2f(\mathbf{x}) + f(\mathbf{y}). \tag{3.11}$$

Với lập luận tương tự, ta cũng có:

$$f(\mathbf{o}_1 \vee \mathbf{y}) - f(\mathbf{y}) \leq f(\mathbf{x}) + 2f(\mathbf{y}).$$

Kết hợp với (3.11), ta được:

$$f(\mathbf{o}_1 \vee \mathbf{x}) + f(\mathbf{o}_1 \vee \mathbf{y}) \leq 4(f(\mathbf{x}) + f(\mathbf{y}))$$

điều này hoàn tất chứng minh. □

Tiếp theo, luận án trình bày kết quả lý thuyết chính của thuật toán FastDrSub trong Định lý 3.3 dưới đây:

Định lý 3.3. *Với một hằng số $\alpha \in (0,1)$, Thuật toán 11 đạt tỉ lệ xấp xỉ là $\frac{1}{8\frac{(2-\alpha)}{1-\alpha}+\frac{1}{\alpha}}$ và có độ phức tạp truy vấn là $O(n\log(k))$. Thuật toán đạt tỉ lệ xấp xỉ $\frac{1}{17+4\sqrt{2}} \approx 0.044$ khi $\alpha = \frac{2\sqrt{2}-1}{7}$.*

Chứng minh. Trước tiên, ta chứng minh tỉ lệ xấp xỉ. Theo quy tắc chọn lựa của thuật toán, $\mathbf{x}'$ là một nghiệm khả thi. Nếu $\|\mathbf{x}\|_1 \leq k$ thì $\mathbf{x}' = \mathbf{x}$. Nếu $\|\mathbf{x}\|_1 > k$, ta có:

$$\begin{aligned} f(\mathbf{x}'_t) - f(\mathbf{x}'_{t-1}) &= f(\mathbf{x}'(e_t)\mathbf{1}_e \mid \mathbf{x}'_{t-1}) \\ &\geq \frac{\mathbf{x}'(e_t) f((\mathbf{x}-\mathbf{x}') \vee \mathbf{x}'_{t-1})}{k} \\ &\geq \frac{\mathbf{x}'(e_t) f(\mathbf{x}-\mathbf{x}')}{k}. \end{aligned}$$

Do quy tắc chọn $\mathbf{x}'$ và $\mathbf{x}$ thỏa mãn $k - \alpha k \leq \|\mathbf{x}'\|_1 \leq k$, ta có:

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{x}-\mathbf{x}') &= \sum_{i=2}^{t} (f(\mathbf{x}'_i) - f(\mathbf{x}'_{i-1})) \geq \sum_{i=2}^{t} \frac{\mathbf{x}'(e_t) f(\mathbf{x}-\mathbf{x}')}{k} \\ &\geq \frac{\|\mathbf{x}'\|_1}{k} f(\mathbf{x}-\mathbf{x}') \geq (1-\alpha) f(\mathbf{x}-\mathbf{x}'). \end{aligned}$$

Từ đó suy ra $f(\mathbf{x}-\mathbf{x}') \leq \frac{f(\mathbf{x})}{2-\alpha}$. Nhờ tính chất submodular, ta có:

$$f(\mathbf{x}) \leq f(\mathbf{x}') + f(\mathbf{x}-\mathbf{x}') \Rightarrow f(\mathbf{x}') \geq f(\mathbf{x}) - f(\mathbf{x}-\mathbf{x}') \geq \frac{1-\alpha}{2-\alpha} f(\mathbf{x}).$$

Tương tự với $\mathbf{y}$, ta thu được $f(\mathbf{y}') \geq \frac{f(\mathbf{y})}{2}$. Áp dụng Bổ đề 3.4, ta có:

$$\begin{aligned} f(\mathbf{o}_1) &\leq f(\mathbf{o}_1 \vee \mathbf{x}) + f(\mathbf{o}_1 \vee \mathbf{y}) \\ &\leq 4(f(\mathbf{x}) + f(\mathbf{y})) \\ &\leq 4\frac{2-\alpha}{1-\alpha}(f(\mathbf{x}') + f(\mathbf{y}')) \\ &\leq 8\frac{2-\alpha}{1-\alpha} f(\mathbf{s}). \end{aligned}$$

Mặt khác, theo định nghĩa của $\mathbf{o}_2$ và quy tắc chọn $(d_{\max}, e_{\max})$ tại dòng 1, ta có:

$$f(\mathbf{o}_2) = f(\sum_{e\in\{\mathbf{o}_2\}} \mathbf{o}_2(e)\mathbf{1}_e) \leq \sum_{e\in\{\mathbf{o}_2\}} f(\mathbf{o}_2(e)\mathbf{1}_e) \tag{3.12}$$

$$\leq \frac{1}{\alpha} f(d_{\max}\mathbf{1}_{e_{\max}}) \leq \frac{f(\mathbf{s})}{\alpha}, \tag{3.13}$$

trong đó bất đẳng thức (3.12) là do tính chất DR-submodular của f. Tổng hợp lại, ta có:

$$f(\mathbf{o}) \leq f(\mathbf{o}_1) + f(\mathbf{o}_2) \leq \left(8\frac{2-\alpha}{1-\alpha} + \frac{1}{\alpha}\right) f(\mathbf{s}).$$

Cuối cùng, ta chứng minh độ phức tạp truy vấn của thuật toán. Để tìm $(e_{\max}, d_{\max})$, với mỗi phần tử e, thuật toán sử dụng phương pháp tìm kiếm nhị phân trên đoạn $[0, \lceil \alpha k \rceil]$ để xác định d_e, với chi phí $n \log(\alpha k)$ truy vấn.

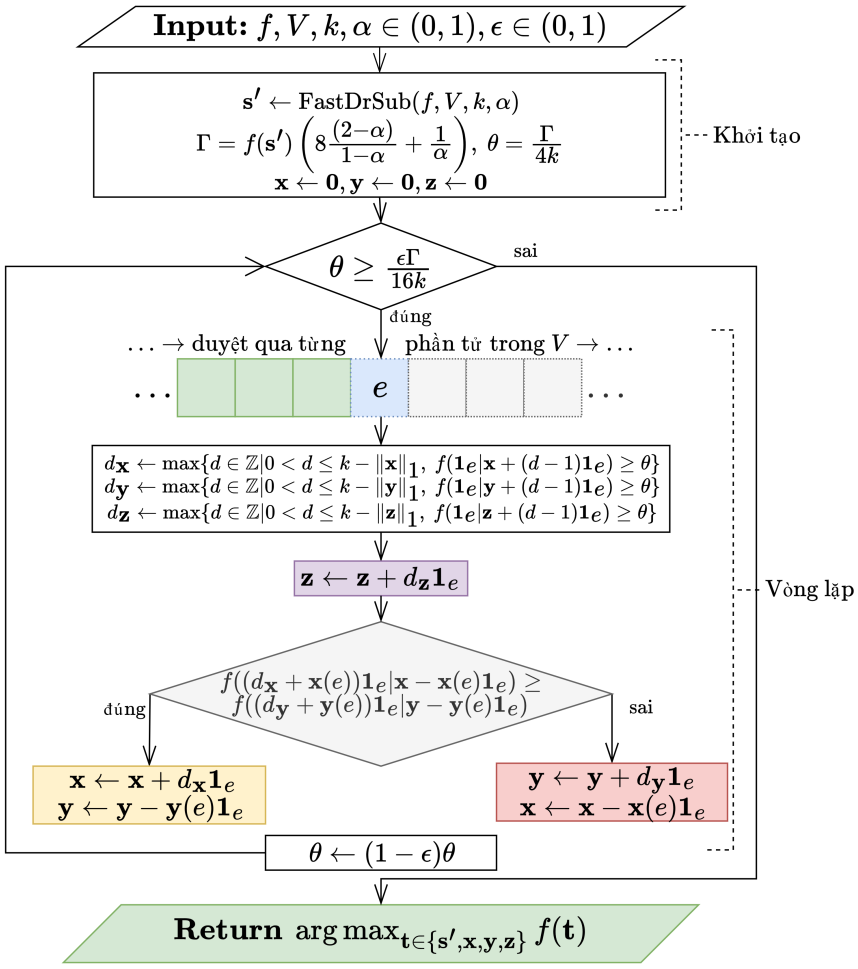
Thuật toán gồm một vòng lặp chính, xử lý từng phần tử e. Với mỗi e, ta áp dụng tìm kiếm nhị phân để xác định $d_{(\mathbf{x},e)}$ và $d_{(\mathbf{y},e)}$, do đó mỗi phần tử yêu cầu $2 \log(\alpha k)$ truy vấn. Tổng cộng số truy vấn là:

$$n \log(k) + 2n \log(\alpha k) = O(n \log k).$$

Điều này hoàn tất chứng minh. □

3.2.3.2. Thuật toán FastDrSub+

Thuật toán FastDrSub+(Thuật toán 12) nhận đầu vào là một hàm DR-submodular f, một tập cơ sở V và các tham số $k \in \mathbb{Z}_+$, $\alpha \in (0, 1)$, $\beta \in (0, 1)$ và $\epsilon \in (0, 1)$. Thuật toán trả về một vector $\mathbf{s} \in \mathbb{Z}_+^V$ sao cho $\|\mathbf{s}\|_1 \leq k$, với giá trị $f(\mathbf{s})$ xấp xỉ giá trị tối ưu của bài toán.



Hình 3.9: Lưu đồ hoạt động của thuật toán FastDrSub+

Ý tưởng cốt lõi của thuật toán là khai thác nghiệm sơ bộ $\mathbf{s}'$ thu được từ thuật toán FastDrSub để ước lượng một cận trên cho giá trị tối ưu, ký hiệu là Γ (Dòng 2) và khởi tạo một ngưỡng chọn lọc $\theta = \frac{\Gamma}{4k\beta}$.

Tiếp theo, thuật toán xây dựng ba vector ứng viên $\mathbf{x}$, $\mathbf{y}$ và $\mathbf{z}$ dựa trên chiến lược ngưỡng giảm dần với hệ số giảm bội $(1-\epsilon)$. Trong quá trình này, hai vector $\mathbf{x}$ và $\mathbf{y}$ được cập nhật luân phiên nhằm đảm bảo các phần tử được chọn thuộc hai tập rời nhau, trong khi vector $\mathbf{z}$ được xây dựng một cách độc lập theo hướng tham lam. Ở mỗi vòng lặp, với mọi phần tử $e \in V$, thuật toán xác định số đơn vị lớn nhất d có thể được thêm vào từng vector sao cho độ lợi biên vẫn không nhỏ hơn θ và cập nhật các vector tương ứng.

Quá trình lặp tiếp tục cho đến khi ngưỡng θ giảm xuống dưới $\frac{\epsilon\Gamma}{16k}$, khi đó thuật toán đánh giá hàm mục tiêu f trên bốn ứng viên: $\mathbf{s}'$, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ và trả về vector ứng viên có giá trị hàm mục tiêu lớn nhất. Cụ thể, các bước được trình bày trong Thuật toán 12 và được minh họa trong Hình 3.9.

Algorithm 12 Thuật toán FastDrSub+

Input: $f : \mathbb{Z}_+^V \to \mathbb{R}_+, V, k, \alpha, \epsilon \in (0,1)$
Output: A solution $\mathbf{s}$ of DrSMC

1: $\mathbf{s}' \leftarrow \text{FastDrSub}(f, V, k, \alpha)$
2: $\Gamma = f(\mathbf{s}')\left(8\frac{(2-\alpha)}{1-\alpha} + \frac{1}{\alpha}\right),\ \theta = \frac{\Gamma}{4k}$
3: $\mathbf{x} \leftarrow \mathbf{0}$, $\mathbf{y} \leftarrow \mathbf{0}$,$\mathbf{z} \leftarrow \mathbf{0}$
4: **while** $\theta \geq \frac{\epsilon\Gamma}{16k}$ **do**
5: **foreach** $e \in V$ **do**
6: $d_{\mathbf{x}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{x}\|_1,\ f(\mathbf{1}_e \mid \mathbf{x} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search
7: $d_{\mathbf{y}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{y}\|_1,\ f(\mathbf{1}_e \mid \mathbf{y} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search
8: $d_{\mathbf{z}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{z}\|_1,\ f(\mathbf{1}_e \mid \mathbf{z} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search
9: $\mathbf{z} \leftarrow \mathbf{z} + d_{\mathbf{z}}\mathbf{1}_e$
10: **if** $f((d_{\boldsymbol{x}} + \boldsymbol{x}(e))\mathbf{1}_e \mid \boldsymbol{x} - \boldsymbol{x}(e)\mathbf{1}_e) \geq f((d_{\boldsymbol{y}} + \boldsymbol{y}(e))\mathbf{1}_e \mid \boldsymbol{y} - \boldsymbol{y}(e)\mathbf{1}_e)$ **then**
11: $\mathbf{x} \leftarrow \mathbf{x} + d_{\mathbf{x}}\mathbf{1}_e$
12: $\mathbf{y} \leftarrow \mathbf{y} - \mathbf{y}(e)\mathbf{1}_e$
13: **else**
14: $\mathbf{y} \leftarrow \mathbf{y} + d_{\mathbf{y}}\mathbf{1}_e$
15: $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{x}(e)\mathbf{1}_e$
16: $\theta \leftarrow (1-\epsilon)\theta$
17: $\mathbf{s} \leftarrow \arg\max_{\mathbf{t} \in \{\mathbf{s}', \mathbf{x}, \mathbf{y}, \mathbf{z}\}} f(\mathbf{t})$
18: **return** $\mathbf{s}$

Luận án tiến hành phân tích và trình bày đảm bảo lý thuyết của thuật toán FastDrSub+ trong Định lý 3.4. Trước hết, ta định nghĩa hoặc nhắc lại một số ký hiệu phục vụ cho việc phân tích trong Bảng 3.7.

Bảng 3.7: Bảng các ký tự toán học dùng trong phân tích thuật toán FastDrSub+

Ký hiệu	Ý nghĩa
$\mathbf{o}$	Nghiệm tối ưu của bài toán DrSMC và $\mathsf{opt} = f(\mathbf{o})$
θ_e	Giá trị của θ tại thời điểm phần tử e được thêm vào vector $\mathbf{x}$ hoặc $\mathbf{y}$
$V_{\mathbf{x}}$	$V_{\mathbf{x}} = \{e \in V : \mathbf{x}(e) > 0, \mathbf{o}(e) = 0\}$
$V_{\mathbf{x}\wedge\mathbf{o}}$	$V_{\mathbf{x}\wedge\mathbf{o}} = \{e \in V : \mathbf{x}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{y}}$	$V_{\mathbf{y}} = \{e \in V : \mathbf{y}(e) > 0, \mathbf{o}(e) = 0\}$
$V_{\mathbf{y}\wedge\mathbf{o}}$	$V_{\mathbf{y}\wedge\mathbf{o}} = \{e \in V : \mathbf{y}(e) \geq \mathbf{o}(e) > 0\}$
$V_{\mathbf{o}}$	$V_{\mathbf{o}} = \{e \in V : \mathbf{o}(e) > 0, \mathbf{o}(e) > \mathbf{x}(e), \mathbf{o}(e) > \mathbf{y}(e)\}$
$\mathbf{x}_e$	$\mathbf{x}_e$ là giá trị của $\mathbf{x}$ ngay trước khi e được thêm vào $\mathbf{x}$
$\mathbf{y}_e$	$\mathbf{y}_e$ là giá trị của $\mathbf{y}$ ngay trước khi e được thêm vào $\mathbf{y}$

Phác thảo chứng minh. FastDrSub+ cải thiện FastDrSub bằng cách dùng nghiệm khởi tạo để ước lượng một cận trên Γ của giá trị tối ưu và sau đó quét các phần tử theo dãy ngưỡng giảm dần. Ba vectơ $\mathbf{x}, \mathbf{y}, \mathbf{z}$ đóng các vai trò khác nhau: $\mathbf{x}$ và $\mathbf{y}$ tạo hai nghiệm ứng viên được cập nhật theo so sánh lợi ích biên, còn $\mathbf{z}$ thu nhận các phần tử vượt ngưỡng để bảo đảm không bỏ sót phần lợi ích lớn. Chứng minh xét hai trường hợp sau vòng lặp: hoặc $\mathbf{z}$ đã đủ lớn để tạo nghiệm tốt, hoặc các phần tử tối ưu còn lại đều có lợi ích biên bị chặn bởi ngưỡng hiện tại. Khi đó, các tổng lợi ích biên được quy về giá trị của $\mathbf{x}, \mathbf{y}, \mathbf{z}$ và nghiệm khởi tạo, từ đó suy ra tỉ lệ $\frac{1}{4} - \epsilon$.

Định lý 3.4. *Với $\epsilon > 0$ và hằng số $\alpha \in (0, 1)$, Thuật toán 12 đạt được tỷ lệ xấp xỉ là $\frac{1}{4} - \epsilon$ và có độ phức tạp truy vấn là $O\left(\frac{n}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\log(k)\right)$.*

Chứng minh. Sau khi vòng lặp $while$ đầu tiên hoàn tất, ta có hai trường hợp:
(1) Nếu $\|\mathbf{x}\|_1 = k$ hoặc $\|\mathbf{y}\|_1 = k$. Giả sử $\|\mathbf{x}\|_1 = k$, theo quy tắc lựa chọn $e \in \{\mathbf{x}\}$, ta có:

$$f(\mathbf{x}) \geq \|\mathbf{x}\|_1 \frac{\Gamma}{4k} \geq \frac{\mathsf{opt}}{4}.$$

Do đó, tỷ lệ xấp xỉ được đảm bảo. Trường hợp $\|\mathbf{y}\|_1 = k$ cũng tương tự.
(2) Nếu $\|\mathbf{x}\|_1 < k$ và $\|\mathbf{y}\|_1 < k$. Ký hiệu θ' là θ tại thời điểm cuối cùng một phần tử e được thêm vào vector $\mathbf{x}$. Xét vector $\mathbf{x}$ tại thời điểm kết thúc vòng lặp $while$, ta phân tích hai trường hợp phụ:

- Nếu $\|\mathbf{x}\|_1 = k$. Giả sử $\theta_{(l)}$ là θ tại thời điểm phần tử cuối cùng được thêm vào $\mathbf{x}$. Khi đó ta có:

$$f(\mathbf{x}) = \sum_{e\in\{x\}} f(\mathbf{x}(e)\mathbf{1}_e|\mathbf{x}_e) \geq k\theta_{(l)}.$$

Mọi $e \in V_{\mathbf{o}}$ không được thêm vào $\mathbf{x}$ đều có lợi ích biên nhỏ hơn θ tại vòng lặp trước đó, tức là $f(\mathbf{1}_e|\mathbf{x}_e) < \frac{\theta_{(l)}}{1-\epsilon}$. Do $\|\mathbf{x}\|_1 = k \geq \|\mathbf{o}\|_1$, ta có:

$$\sum_{e\in V_{\mathbf{x}}} \mathbf{x}(e) + \sum_{e\in V_{\mathbf{x}\wedge\mathbf{o}}} \mathbf{x}(e) + \sum_{e\in V_{\mathbf{o}}} \mathbf{x}(e) = k \geq \sum_{e\in V_{\mathbf{o}}} \mathbf{x}(e) + \sum_{e\in V_{\mathbf{x}\wedge\mathbf{o}}} \mathbf{o}(e) + \sum_{e\in V_{\mathbf{o}}} t_e$$

điều này dẫn đến $\sum_{e\in V_{\mathbf{x}}} \mathbf{x}(e) \geq \sum_{e\in V_{\mathbf{o}}} t_e$. Kết hợp lại, ta có:

$$\begin{aligned}
\sum_{e\in V_{\mathbf{o}}} f(t_e\mathbf{1}_e|\mathbf{x}) &\leq \sum_{e\in V_{\mathbf{o}}} t_e f(\mathbf{1}_e|\mathbf{x}_e) \ \text{(Theo Bổ đề 1.2)} \\
&\leq \sum_{e\in V_{\mathbf{o}}} \frac{t_e\theta_{(l)}}{1-\epsilon} = \frac{\theta_{(l)}}{1-\epsilon}\sum_{e\in V_{\mathbf{o}}} t_e \\
&\leq \frac{\theta_{(l)}}{1-\epsilon}\sum_{e\in V_{\mathbf{x}}} \mathbf{x}(e) = \sum_{e\in V_{\mathbf{x}}} \frac{\mathbf{x}(e)\theta_{(l)}}{1-\epsilon} \\
&\leq \sum_{e\in V_{\mathbf{x}}} \frac{\mathbf{x}(e)\theta_e}{1-\epsilon} \\
&\leq \sum_{e\in V_{\mathbf{x}}} \sum_{d=1}^{\mathbf{x}(e)} \frac{f(\mathbf{1}_e|\mathbf{x}_e + (d-1)\mathbf{1}_e)}{1-\epsilon} \\
&\leq \sum_{e\in V_{\mathbf{x}}} \frac{f(\mathbf{x}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon}.
\end{aligned}$$

- Nếu $\|\mathbf{x}\|_1 < k$. Trong trường hợp này, mọi $e \in V_{\mathbf{o}}$ không được thêm vào $\mathbf{x}$ đều có lợi ích biên nhỏ hơn θ tại vòng lặp cuối. Do đó $f(\mathbf{1}_e|\mathbf{x}_e) < \frac{\epsilon M}{k}$ và suy ra:

$$\sum_{e\in V_{\mathbf{o}}} f(t_e\mathbf{1}_e|\mathbf{x}) \leq \sum_{e\in V_{\mathbf{o}}} t_e\frac{\epsilon M}{k} \leq \epsilon M \leq \epsilon\mathsf{opt}.$$

Kết hợp hai trường hợp, ta có:

$$\sum_{e\in V_{\mathbf{o}}} f(t_e\mathbf{1}_e|\mathbf{x}) \leq \epsilon\mathsf{opt} + \sum_{e\in V_{\mathbf{x}}} \frac{f(\mathbf{x}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon}.$$

Tương tự, ta cũng có thuộc tính tương tự với $\mathbf{y}$:

$$\sum_{e\in V_{\mathbf{o}}} f(t_e\mathbf{1}_e|\mathbf{y}) \leq \epsilon\mathsf{opt} + \sum_{e\in V_{\mathbf{y}}} \frac{f(\mathbf{y}(e)\mathbf{1}_e|\mathbf{y}_e)}{1-\epsilon}$$

Với $e \in V_{\mathbf{o}\wedge\mathbf{y}}$, giả sử $d_{(\mathbf{y},e)}\mathbf{1}_e$ được thêm vào $\mathbf{y}$ tại vòng lặp j và không được thêm vào $\mathbf{y}$ ở vòng trước đó, ta có $f(\mathbf{1}_e|\mathbf{x}) \leq \frac{\theta_e}{1-\epsilon}$. Do đó,

$$f(\mathbf{y}(e)\mathbf{1}_e|\mathbf{x}) = f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{x}) < \frac{d_{(\mathbf{y},e)}\theta_e}{1-\epsilon} \leq \frac{f(d_{(\mathbf{y},e)}\mathbf{1}_e|\mathbf{y}_e)}{1-\epsilon} = \frac{f(\mathbf{y}(e)\mathbf{1}_e|\mathbf{y}_e)}{1-\epsilon}.$$

Kết hợp lại, ta phân tích mối quan hệ giữa $f(\mathbf{o}\vee\mathbf{x})$ và $f(\mathbf{x})$ như sau:

$$\begin{aligned}
f(\mathbf{o}\vee\mathbf{x}) - f(\mathbf{x}) &\leq \sum_{e\in V_{\mathbf{o}}} f(t_e\mathbf{1}_e \mid \mathbf{x}) + \sum_{e\in V_{\mathbf{o}\wedge\mathbf{y}}} f(\mathbf{y}(e)\mathbf{1}_e \mid \mathbf{x}) \\
&\leq \epsilon\mathsf{opt} + \sum_{e\in V_{\mathbf{x}}} \frac{f(\mathbf{x}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon} + \sum_{e\in V_{\mathbf{o}\wedge\mathbf{y}}} f(\mathbf{y}(e)\mathbf{1}_e \mid \mathbf{y}_e). \qquad (3.14)
\end{aligned}$$

Tương tự, ta có thuộc tính tương tự với $\mathbf{y}$:

$$f(\mathbf{o}\vee\mathbf{y}) - f(\mathbf{y}) \leq \epsilon\mathsf{opt} + \sum_{e\in V_{\mathbf{y}}} \frac{f(\mathbf{y}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon} + \sum_{e\in V_{\mathbf{o}\wedge\mathbf{x}}} f(\mathbf{x}(e)\mathbf{1}_e \mid \mathbf{x}_e). \qquad (3.15)$$

Kết hợp (3.14) và (3.15), ta suy ra:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{x}) - f(\mathbf{y}) &\leq f(\mathbf{o} \vee \mathbf{x}) - f(\mathbf{x}) + f(\mathbf{o} \vee \mathbf{y}) - f(\mathbf{y}) \\ &\leq \sum_{e \in V_{\mathbf{x}}} \frac{f(\mathbf{x}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon} + \sum_{e \in V_{\mathbf{o} \wedge \mathbf{x}}} f(\mathbf{x}(e)\mathbf{1}_e \mid \mathbf{x}_e) \\ &+ \sum_{e \in V_{\mathbf{y}}} \frac{f(\mathbf{y}(e)\mathbf{1}_e|\mathbf{x}_e)}{1-\epsilon} + \sum_{e \in V_{\mathbf{o} \wedge \mathbf{y}}} f(\mathbf{y}(e)\mathbf{1}_e \mid \mathbf{y}_e) \\ &\leq \frac{f(\mathbf{x}) + f(\mathbf{y})}{1-\epsilon}. \end{aligned}$$

Do đó,

$$f(\mathbf{s}) \geq \frac{f(\mathbf{x}) + f(\mathbf{y})}{2} \geq \frac{1-\epsilon}{4-2\epsilon} f(\mathbf{o}) = (\frac{1}{4} - \frac{\epsilon}{2(4-2\epsilon)})\mathsf{opt} \geq (\frac{1}{4} - \epsilon)\mathsf{opt}.$$

Về độ phức tạp truy vấn, vòng lặp $while$ có độ phức tạp là $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$. Vòng lặp **for** lồng trong có độ phức tạp là $O(n\log(k))$. Do đó, độ phức tạp tổng thể của thuật toán là $O(\frac{n}{\epsilon}\log(\frac{1}{\epsilon})\log(k))$. Chứng minh hoàn tất. □

3.2.4. Thực nghiệm

Trong mục này, luận án trình bày các thực nghiệm nhằm đánh giá hiệu quả của các thuật toán được đề xuất trong việc giải bài toán DrSMC. Do hiện chưa có nghiên cứu nào trực tiếp xử lý bài toán này, luận án áp dụng kỹ thuật ánh xạ theo đề xuất của Ene và Nguyen [34] để chuyển bài toán DrSMC về dạng bài toán SMK. Cách tiếp cận này cho phép tiến hành so sánh với hai thuật toán hiện đại tiêu biểu trong bài toán SMK, bao gồm RLA và SMKRANACC. Các thuật toán được so sánh dựa trên ba tiêu chí chính: (1) giá trị hàm mục tiêu đạt được, (2) số lượng truy vấn đến hàm mục tiêu f và (3) thời gian thực thi. Thực nghiệm được tiến hành với ứng dụng tối đa doanh thu trên các bộ dữ liệu chuẩn đã được công bố rộng rãi nhằm bảo đảm tính khách quan và khả năng tái lập. Việc thiết lập như vậy cho phép đặt các phương pháp đề xuất trong mối tương quan trực tiếp với các chuẩn hiện hành. Kết quả tổng hợp cho thấy FastDrSub+ đạt chất lượng nghiệm cùng nhóm với RLA và SMKRANACC, trong khi FastDrSub có lợi thế rõ về số lượng truy vấn và thời gian chạy.

3.2.4.1. Ứng dụng và bộ dữ liệu

Dựa trên ứng dụng Tối đa doanh thu *(Định nghĩa 1.26)* sử dụng hàm submodular, luận án xây dựng một mô hình tương tự áp dụng hàm DR-submodular như sau: Xét một mạng xã hội vô hướng được biểu diễn dưới dạng đồ thị $G = (V, E)$, trong đó V là tập người dùng (các đỉnh) và E là tập các kết nối (các cạnh) giữa họ. Mỗi cạnh $(u, v) \in E$ được gán một trọng số không âm $w_{uv} \in [0, 1]$ biểu thị mức độ ảnh hưởng từ người dùng u tới người dùng v. Giả định ngân sách quảng cáo có giới hạn và cần được phân bổ sao cho tối đa khả năng tiếp cận sản phẩm trong mạng.

Với mỗi người dùng u, một khoản đầu tư $\mathbf{x}(u)$ biểu thị mức độ nỗ lực quảng bá sản phẩm thông qua người dùng đó. Tập người dùng nhận được đầu tư là $\{\mathbf{x}\} = \{u \in V \mid \mathbf{x}(u) > 0\}$. Mục tiêu là tối đa doanh thu kỳ vọng, tức là số lượng người dùng sẽ tiếp nhận sản phẩm dựa trên ảnh hưởng từ các người dùng trong tập $\{\mathbf{x}\}$ đến phần còn lại của mạng.

Hàm doanh thu $f(\mathbf{x})$ được định nghĩa như sau:

$$f(\mathbf{x}) = \sum_{u \in V \setminus \{\mathbf{x}\}} f_u\Big(\sum_{v \in \{\mathbf{x}\}} w_{uv}\mathbf{x}(v)\Big),$$

trong đó:

- $f_u(t) = \log(1 + t^{\alpha_u})$ là một hàm lõm, không âm, biểu diễn xác suất người dùng u tiếp nhận sản phẩm dưới ảnh hưởng tích lũy $t = \sum_{v \in \{\mathbf{x}\}} w_{uv}\mathbf{x}(v)$. Tham số $\alpha_u \in (0, 1)$ là hệ số cá nhân hóa cho mỗi người dùng, phản ánh hiện tượng bão hòa trong tác động từ tập $\{\mathbf{x}\}$ đến u.
- w_{uv} biểu thị mức độ ảnh hưởng từ đỉnh u đến đỉnh v.

Hàm doanh thu $f(\mathbf{x})$ là một hàm DR-submodular. Điều này được chứng minh dựa trên tính lõm của $f_u(t)$, theo đó mức tăng thêm trong $f_u(t)$ sẽ giảm khi t tăng. Khi thêm một phần tử vào tập người dùng được đầu tư $\{\mathbf{x}\}$, tác động đến doanh thu sẽ lớn hơn nếu vector $\mathbf{x}$ có ít phần tử hơn (tức ít người dùng đã được đầu tư). Điều này phản ánh hiện tượng lợi ích biên giảm dần trong ảnh hưởng và thỏa mãn bất đẳng thức DR-submodular:

$$f(\mathbf{x} + \mathbf{1}_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \mathbf{1}_e) - f(\mathbf{y})$$

với mọi $\mathbf{x} \leq \mathbf{y}$. Do đó, hàm $f(\mathbf{x})$ là DR-submodular.

Để đánh giá hiệu năng của các thuật toán, luận án sử dụng ba bộ dữ liệu thực nghiệm: Facebook, AstroPh và Enron. Các bộ dữ liệu này có quy mô khác nhau: Facebook là nhỏ nhất với $4,039$ đỉnh; Enron là lớn nhất với $36,692$ đỉnh; trong khi AstroPh có $18,772$ đỉnh, nằm ở mức trung bình. Các bộ dữ liệu chuẩn này được thu thập từ SNAP (xem Bảng 3.8).

Bảng 3.8: Bảng thống kê các bộ dữ liệu sử dụng cho thực nghiệm trong bài toán DrSMC

Bộ dữ liệu	**Đỉnh**	**Cạnh**	**Phân loại**	**Nguồn**
Facebook	$4,039$	$88,234$	Vô hướng	SNAP
AstroPh	$18,772$	$198,110$	Vô hướng	SNAP
Enron	$36,692$	$183,831$	Vô hướng	SNAP

3.2.4.2. Thuật toán so sánh

Như đã trình bày ở trên, hiện chưa có công trình nào nghiên cứu trực tiếp bài toán DrSMC. Do đó, luận án tiến hành đánh giá hiệu quả của các thuật toán đề xuất

bằng cách so sánh với các phương pháp tiên tiến trong bài toán SMK, cụ thể là RLA và SMKRANACC. Thông tin chi tiết về các thuật toán được so sánh như sau:

- **FastDrSub**: Tỷ lệ xấp xỉ 0.044 với độ phức tạp truy vấn $O(n\log(k))$.
- **FastDrSub+**: Tỷ lệ xấp xỉ $\frac{1}{4}-\epsilon$ với độ phức tạp truy vấn $O\left(\frac{n}{\epsilon}\log\left(\frac{k}{\epsilon}\right)\right)$.
- **RLA:** Tỷ lệ xấp xỉ $\frac{1}{4}-\epsilon$ với độ phức tạp truy vấn $O(\frac{n(\log k+\frac{1}{\epsilon})}{\epsilon}\log\frac{1}{\epsilon})$.
- **SMKRANACC [53]:** Tỷ lệ xấp xỉ $\frac{1}{4}-\epsilon$ với độ phức tạp truy vấn $O(\frac{n(\log k+\frac{1}{\epsilon})}{\epsilon}\log\frac{k}{\epsilon})$.

Để mở rộng khả năng áp dụng của RLA và SMKRANACC sang bài toán DR-submodular, luận án thực hiện điều chỉnh cần thiết theo kỹ thuật ánh xạ được đề xuất bởi Ene [34]. Các điều chỉnh này vẫn bảo toàn nguyên lý hoạt động cốt lõi và các bảo đảm lý thuyết của các thuật toán gốc.

Thiết lập tham số. Thuật toán FastDrSub được thực thi với nhiều giá trị khác nhau của tham số α thuộc tập $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, được ký hiệu trong các biểu đồ tương ứng là FastDrSub-0.1, ..., FastDrSub-0.9. Đối với thuật toán FastDrSub+, các tham số được cố định là $\alpha = \frac{2\sqrt{2}-1}{7} \approx 0.2612$ và $\beta = 0.2$. Tất cả các thuật toán đều sử dụng giá trị tham số chung là $\epsilon = 0.1$. Các thực nghiệm được triển khai trên cụm máy chủ hiệu năng cao (HPC) với cấu hình như sau: phân vùng = large, số luồng CPU = 16, số nút xử lý = 2 và bộ nhớ tối đa = 3,073 GB.

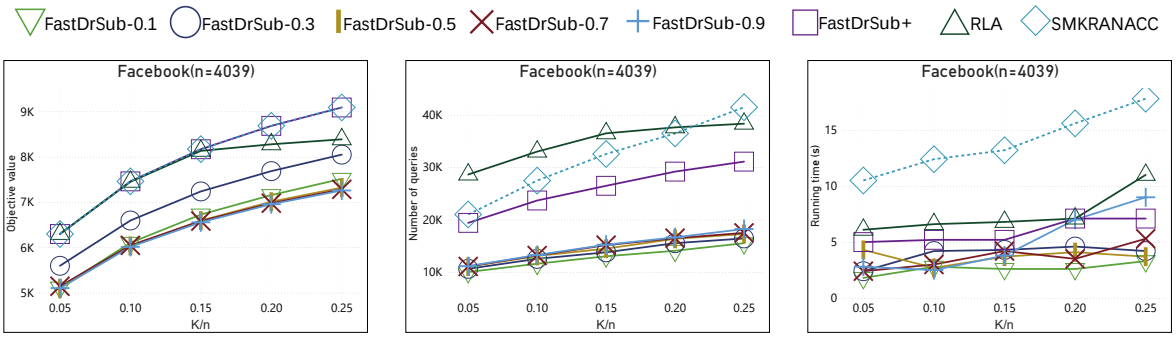
Bảng 3.9: Thiết lập thực nghiệm cho bài toán DrSMC

Nhóm thuật toán	**Thuật toán**	**Vai trò so sánh**	**Tham số**	**Tiêu chí**
Đề xuất	FastDrSub, FastDrSub+	Đánh giá trực tiếp trên bài toán DrSMC	FastDrSub: $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$; FastDrSub+: $\alpha \approx 0.2612$, $\beta = 0.2$	Giá trị, truy vấn, thời gian
Đối sánh qua ánh xạ	RLA, SMKRANACC	Các thuật toán SMK được áp dụng thông qua kỹ thuật rút gọn/ánh xạ	$\epsilon = 0.1$	Giá trị, truy vấn, thời gian

3.2.4.3. Kết quả thực nghiệm

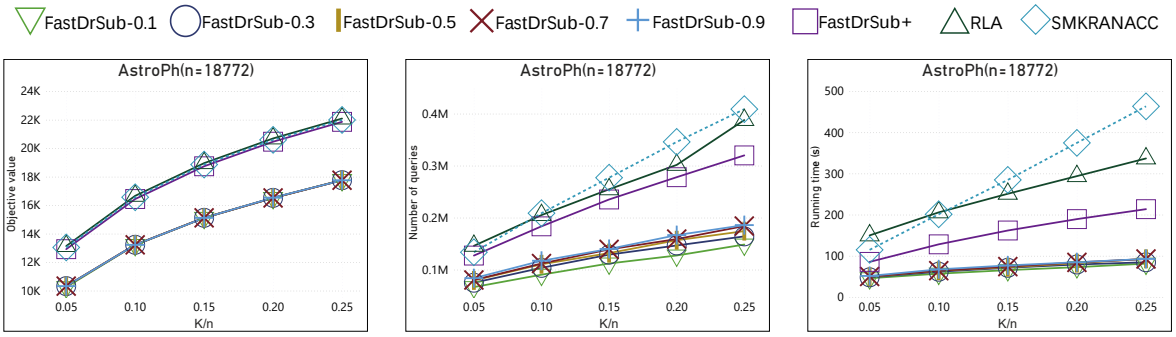
Kết quả về giá trị hàm mục tiêu được trình bày trong Hình 3.10a, 3.11a và 3.12a. Các thuật toán được chia thành hai nhóm dựa trên giá trị hàm mục tiêu. Nhóm có giá trị hàm mục tiêu cao bao gồm FastDrSub+, RLA và SMKRANACC, trong khi nhóm có giá trị thấp hơn bao gồm FastDrSub với các tham số α khác nhau.

Giá trị hàm mục tiêu của nhóm hiệu năng cao lớn hơn nhóm hiệu năng thấp khoảng 1.2 đến 1.4 lần. Trong nhóm hiệu năng cao, các thuật toán đạt giá trị hàm mục tiêu tương đương nhau, với sai khác nhỏ không mang ý nghĩa thống kê trên cả ba bộ dữ liệu. Cụ thể, với bộ dữ liệu Facebook, thuật toán RLA có kết quả thấp hơn so với FastDrSub+



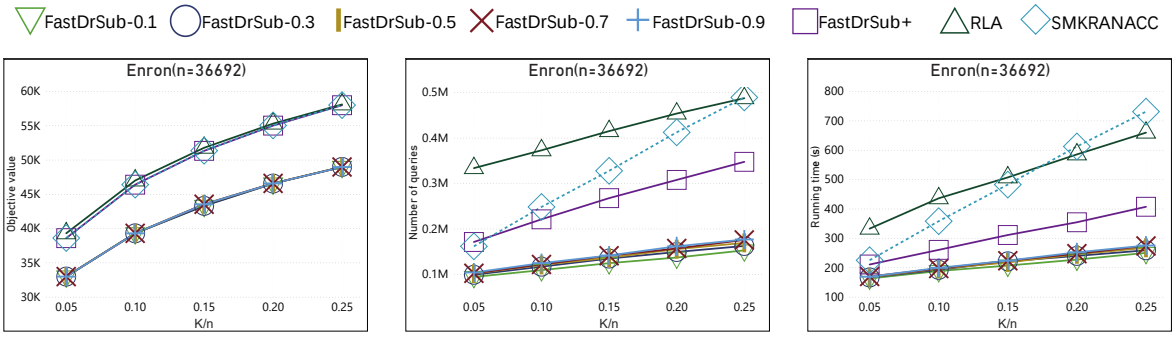
(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.10: Kết quả thực nghiệm trên ứng dụng tối đa doanh thu cho bài toán DrSMC với tập dữ liệu Facebook.



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.11: Kết quả thực nghiệm trên ứng dụng tối đa doanh thu cho bài toán DrSMC với tập dữ liệu Astro.



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn (c) Thời gian chạy

Hình 3.12: Kết quả thực nghiệm trên ứng dụng tối đa doanh thu cho bài toán DrSMC với tập dữ liệu Enron.

và SMKRANACC tại các điểm 0.2 và 0.25. Đối với nhóm hiệu năng thấp, bao gồm FastDrSub với các giá trị α khác nhau, giá trị hàm mục tiêu gần như tương đương nhau trên hai bộ dữ liệu lớn là AstroPh và Enron. Tuy nhiên, với bộ dữ liệu Facebook, FastDrSub với $\alpha = 0.3$ đạt giá trị hàm mục tiêu cao nhất, cao hơn các giá trị α khác

khoảng 1.1 đến 1.2 lần trong thiết lập này. Ngược lại, $\alpha = 0.9$ cho kết quả thấp nhất, tuy nhiên mức chênh lệch là không đáng kể.

Kết quả về số lượng truy vấn được thể hiện trong Hình 3.10b, 3.11b và 3.12b. Tương tự như giá trị hàm mục tiêu, số truy vấn của các thuật toán cũng được chia thành hai nhóm. Nhóm có số lượng truy vấn cao hơn bao gồm FastDrSub+, RLA và SMKRANACC; nhóm có số lượng truy vấn thấp hơn bao gồm FastDrSub với các giá trị α khác nhau. Trên cả ba bộ dữ liệu, số lượng truy vấn của FastDrSub khá ổn định, với chênh lệch nhỏ giữa các thiết lập α. Đáng chú ý, FastDrSub+ tạo ra số lượng truy vấn ít nhất trên bộ dữ liệu Facebook so với RLA và SMKRANACC, thấp hơn RLA khoảng 1.2 lần. Xu hướng tương tự cũng được quan sát trên bộ dữ liệu Enron.

Cuối cùng, Hình 3.10c, 3.11c và 3.12c trình bày kết quả về thời gian thực thi. Thời gian thực thi của các thuật toán có xu hướng tương tự với số lượng truy vấn do mối quan hệ trực tiếp giữa hai đại lượng này. Thuật toán nào thực hiện nhiều truy vấn hơn sẽ có thời gian chạy dài hơn. Nhìn chung, FastDrSub với các giá trị α khác nhau có thời gian thực thi ngắn nhất trên cả ba bộ dữ liệu, với sai khác nhỏ và không đáng kể giữa các thiết lập α. Thuật toán FastDrSub+ có thời gian chạy cao hơn FastDrSub nhưng thấp hơn so với RLA và SMKRANACC trong tất cả các trường hợp. Cụ thể, trên hai bộ dữ liệu AstroPh và Enron, thời gian chạy của FastDrSub+ nhanh hơn từ 1.5 đến 2 lần so với RLA và SMKRANACC.

Bảng 3.10: Tổng hợp nhận xét chính từ thực nghiệm cho bài toán DrSMC

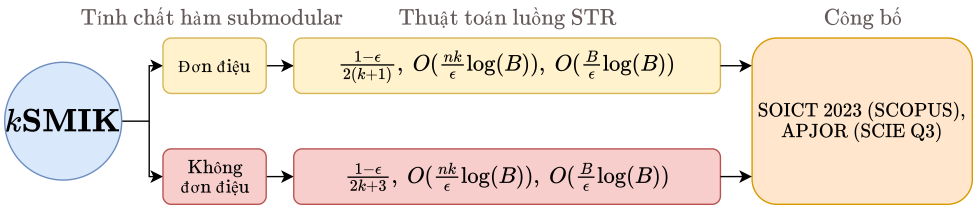
Nhóm thuật toán	**Giá trị hàm mục tiêu**	**Số truy vấn**	**Thời gian chạy**
FastDrSub+, RLA, SMKRANACC	Thuộc nhóm giá trị cao và khác biệt nhỏ trên ba bộ dữ liệu.	FastDrSub+ thường cần ít truy vấn hơn RLA và SMKRANACC ở một số bộ dữ liệu.	FastDrSub+ có thời gian chạy thấp hơn RLA và SMKRANACC trong các thiết lập được khảo sát.
FastDrSub	Giá trị thấp hơn nhóm FastDrSub+/RLA/SMKRANACC, phụ thuộc vào α.	Số truy vấn thấp và ổn định nhất.	Nhanh nhất trong các thuật toán so sánh, phù hợp khi ưu tiên chi phí tính toán.

Bảng 3.10 cho thấy hai thuật toán đề xuất có vai trò bổ sung cho nhau. FastDrSub+ phù hợp khi mục tiêu chính là đạt chất lượng nghiệm cao trong khi vẫn giảm thời gian chạy so với các thuật toán SMK được ánh xạ. Ngược lại, FastDrSub phù hợp với các tình huống cần giảm mạnh số lượng truy vấn và thời gian thực thi, chấp nhận đánh đổi một phần chất lượng nghiệm. Do các thuật toán RLA và SMKRANACC không được thiết kế trực tiếp cho DrSMC mà được sử dụng thông qua kỹ thuật ánh xạ, kết quả thực nghiệm cần được hiểu như một đối sánh tham khảo với các chuẩn gần nhất hiện có, thay vì so sánh trực tiếp với một thuật toán chuyên biệt cho cùng bài toán.

3.3. Kết luận chương

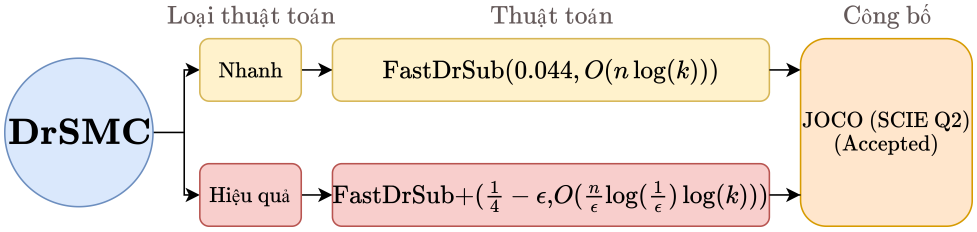
Chương này trình bày hai hướng nghiên cứu chính trong tối ưu hàm submodular mở rộng: (i) tối ưu hàm k-submodular với ràng buộc chi phí nhóm và (ii) tối ưu hàm DR-submodular với ràng buộc lực lượng. Các bài toán này phản ánh nhu cầu thực tế trong nhiều ứng dụng như phân phối cảm biến, tối đa ảnh hưởng đa chủ đề và phân bổ tài nguyên trong hệ thống học máy. Các thuật toán được đề xuất đều có khả năng xử lý dữ liệu lớn với đảm bảo về hiệu quả xấp xỉ và chi phí tính toán thấp.

Đối với bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm, luận án đề xuất thuật toán luồng STR, chỉ cần hai lượt quét qua dữ liệu và sử dụng bộ nhớ giới hạn. Thuật toán này đạt hệ số xấp xỉ $\frac{1-\epsilon}{2(k+1)}$ trong trường hợp đơn điệu và $\frac{1-\epsilon}{2k+3}$ trong trường hợp không đơn điệu, với độ phức tạp truy vấn là $O(\frac{nk}{\epsilon}\log(B))$ và bộ nhớ $O(\frac{B}{\epsilon}\log(n))$ (xem Hình 3.13). Trong các thực nghiệm đã khảo sát, STR thường đạt chất lượng nghiệm cạnh tranh với phương pháp tham lam và giảm đáng kể số lần truy vấn hàm mục tiêu, phù hợp với môi trường dữ liệu lớn.



Hình 3.13: Tổng hợp các kết quả chính của luận án cho bài toán kSMIK.

Với bài toán tối đa hàm DR-submodular với ràng buộc lực lượng, luận án đề xuất hai thuật toán mới là FastDrSubvà FastDrSub+, lần lượt đạt hệ số xấp xỉ 0.044 và $\frac{1}{4}-\epsilon$ với độ phức tạp truy vấn $O(n\log(k))$ (xem Hình 3.14). Theo đối sánh trong chương, các thuật toán này đạt hệ số xấp xỉ hằng số với chi phí truy vấn thấp cho bài toán DrSMC không đơn điệu. Các thực nghiệm được triển khai trên bài toán tối đa doanh thu với hàm mục tiêu dạng DR-submodular cho thấy FastDrSub+ đạt chất lượng nghiệm cạnh tranh với RLA và SMKRANACC, trong khi FastDrSub có lợi thế về số lượng truy vấn và thời gian chạy.



Hình 3.14: Tổng hợp các kết quả chính của luận án cho bài toán DrSMC.

Nhìn tổng thể, Chương 3 đóng vai trò mở rộng hướng nghiên cứu từ bài toán SMK nền tảng ở Chương 2 sang các mô hình có cấu trúc phong phú hơn: ràng buộc

chi phí nhóm trong không gian k-submodular và ràng buộc lực lượng trong không gian DR-submodular. Mạch này chuẩn bị cho chương cuối, nơi luận án tiếp tục xét bài toán MSC như một hướng đối ngẫu/mở rộng khác của bài toán nền. Các kết quả lý thuyết và thực nghiệm trong chương này không chỉ góp phần mở rộng hiểu biết về tối ưu submodular mở rộng mà còn tạo tiền đề cho việc áp dụng các thuật toán hiệu quả vào các hệ thống học máy và ra quyết định tự động. Đặc biệt, các kết quả trong chương đã được công bố tại các hội nghị và tạp chí quốc tế uy tín, qua đó thể hiện giá trị học thuật và khả năng ứng dụng thực tiễn của các hướng tiếp cận được phát triển trong luận án:

1. **Tan D. Tran**, Canh V. Pham, Dung K.T Ha, *Maximizing a k-submodular Maximization Function under an Individual Knapsack Constraint*, 2023, In Proceedings of the 12th International Symposium on Information and Communication Technology (SOICT 2023), 56-62 **(SCOPUS)**;

2. **Tan D. Tran**, Canh V. Pham, Dung K.T Ha, *k-submodular Maximization Under Individual Knapsack Constraints: Applications and Streaming Algorithm*, 2025, Asia-Pacific Journal of Operational Research **(SCIE, Q3)**.

3. **Tan D. Tran**, Canh V. Pham, *Fast Approximation Algorithm for Non-Monotone DR-submodular Maximization under Size Constraint*, 2025, Journal of Combinatorial Optimization **(SCIE, Q2)**.

CHƯƠNG 4
BÀI TOÁN PHỦ SUBMODULAR VỚI CHI PHÍ TỐI THIỂU

Chương này, luận án tập trung vào bài toán phủ submodular với chi phí tối thiểu (Minimum cost submodular Cover – viết tắt là MSC) – Bài toán nghiên cứu 4. Bài toán MSC là đối ngẫu của bài toán tối đa hàm submodular với ràng buộc chi phí, là một trong những bài toán cơ bản trong tối ưu hàm submodular, có nhiều ứng dụng trong thực tế. Nếu Chương 2 nghiên cứu cách tối đa hóa giá trị hàm mục tiêu khi ngân sách bị giới hạn, thì chương này xét chiều ngược lại: tìm chi phí nhỏ nhất để đạt một ngưỡng giá trị cho trước. Vì vậy, MSC khép lại mạch nghiên cứu của luận án bằng cách chuyển từ bài toán tối đa dưới ràng buộc chi phí sang bài toán tối thiểu chi phí dưới ràng buộc bao phủ. Đóng góp chính của chương là đề xuất và phân tích ba thuật toán luồng xấp xỉ hai tiêu chí cho bài toán MSC, đồng thời đối sánh chúng với các thuật toán hiện có trong cùng lớp bài toán:

- **SingStr**: Thuật toán một lượt quét đạt hệ số xấp xỉ $\left(\frac{(1+\epsilon)(2-\epsilon)}{\epsilon}, 1-\epsilon\right)$, độ phức tạp truy vấn $O\left(\frac{n}{\epsilon}\log\left(\frac{\rho n}{\epsilon}\right)\right)$ và độ phức tạp bộ nhớ $O\left(\frac{k}{\epsilon}\log\left(\frac{\rho n}{\epsilon}\right)\right)$, trong đó n là kích thước dữ liệu đầu vào, k là kích thước lời giải, ϵ là tham số chính xác, $\rho = \frac{c_{\max}}{c_{\min}}$ biểu thị tỉ lệ giữa chi phí phần tử lớn nhất $(c_{\max})$ và chi phí phần tử nhỏ nhất $(c_{\min})$.

- **ThreeStr**: Thuật toán ba lượt quét đạt cùng hệ số xấp xỉ như SingStr nhưng cải thiện độ phức tạp truy vấn và bộ nhớ xuống $O\left(\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$ và $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$. Theo đối sánh trong Bảng 4.1, đây là một phương án luồng ba lượt quét có độ phức tạp đa thức và không phụ thuộc trực tiếp vào các tham số bất lợi như $c_{\min}$ trong độ phức tạp truy vấn.

- **MultiStr**: Thuật toán nhiều lượt quét hằng số $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$, đạt hệ số xấp xỉ $\left(\left(1+\ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon}, 1-\epsilon\right)$. Thuật toán này hướng tới sự đánh đổi giữa số lượt quét và chất lượng nghiệm, trong đó hệ số xấp xỉ tiệm cận gần hơn với phương pháp tham lam so với các thuật toán luồng một lượt/quy mô lớn đang được đối sánh.

Ba thuật toán được thiết kế phù hợp với các kịch bản xử lý dữ liệu lớn theo luồng. Các kết quả nghiên cứu tương ứng đã được công bố tại The 12th International Conference on Computational Data and Social Networks (CSoNet 2023), kỷ yếu hội nghị được lập chỉ mục trong SCOPUS. Cấu trúc của chương được tổ chức như sau. Trước tiên, luận án trình bày các khái niệm, ký hiệu liên quan đến bài toán và tổng quan các nghiên cứu liên quan đến MSC. Tiếp theo, ba thuật toán đề xuất được giới thiệu chi tiết, bao gồm phân tích lý thuyết về hệ số xấp xỉ, độ phức tạp truy vấn và bộ nhớ. Phần đánh giá thực nghiệm trình bày hiệu quả các thuật toán trên hai ứng dụng thực nghiệm, so sánh với các phương pháp hiện có. Cuối cùng là phần kết luận chương.

4.1. Mô tả bài toán

4.1.1. Định nghĩa bài toán

Trong Chương 2, luận án đã trình bày bài toán tối đa hàm submodular với ràng buộc chi phí, trong đó mục tiêu là chọn tập con sao cho tổng chi phí không vượt quá ngân sách cho trước, đồng thời giá trị hàm mục tiêu đạt được là lớn nhất. Tuy nhiên, trong nhiều ứng dụng thực tế, yêu cầu tối đa trong nhiều trường hợp là không cần thiết hoặc không khả thi, mà giá trị hàm mục tiêu chỉ cần đạt đến một ngưỡng cho trước. Khi đó, mục tiêu phù hợp hơn là tối thiểu chi phí để đạt được hiệu quả tối thiểu yêu cầu. Bài toán phủ submodular mô tả chính xác tình huống này: cho một ngưỡng $T > 0$, bài toán yêu cầu tìm một tập $S \subseteq V$ sao cho $f(S) \geq T$. Trong mô hình phủ submodular với chi phí tối thiểu, mỗi phần tử $e \in V$ gắn với chi phí $c(e)$ và mục tiêu là tìm tập S có chi phí nhỏ nhất thỏa mãn $f(S) \geq T$. Bài toán phủ submodular với chi phí tối thiểu có thể được xem như "đối ngẫu" tự nhiên của bài toán tối đa hàm submodular với ràng buộc chi phí: thay vì tối đa giá trị hàm mục tiêu thu được dưới một giới hạn chi phí, ta tìm chi phí nhỏ nhất để đạt được một giá trị hàm mục tiêu cho trước. Lúc này, mục tiêu của bài toán là tối thiểu tổng chi phí, hàm submodular trở thành ràng buộc của bài toán.

Một ví dụ tiêu biểu của bài toán phủ submodular với chi phí tối thiểu là trong các chiến dịch tiếp thị trên mạng xã hội. Giả sử mục tiêu của chiến dịch là lan tỏa thông tin đến ít nhất một tỷ lệ người dùng nhất định trong mạng. Khi đó, thay vì tìm tập người ảnh hưởng tối đa tổng ảnh hưởng (có thể tốn kém và không cần thiết), bài toán đặt ra là tìm tập người ảnh hưởng có chi phí thấp nhất mà vẫn bảo đảm lan tỏa thông tin đến một ngưỡng yêu cầu. Mô hình này thể hiện rõ tính tiết kiệm tài nguyên, phù hợp với các chiến lược hoạt động "đủ tốt" trong điều kiện ngân sách hạn chế.

Cụ thể, bài toán phủ submodular với chi phí tối thiểu được phát biểu như sau:

Định nghĩa 4.1 (Bài toán phủ submodular với chi phí tối thiểu – MSC)**.** Cho một tập cơ sở $V = \{e_1, \ldots, e_n\}$, một hàm submodular đơn điệu $f : 2^V \to \mathbb{R}_+$ với $f(\emptyset) = 0$ và mỗi phần tử $e \in V$ có chi phí $c(e) > 0$. Chi phí của một tập con $S \subseteq V$ là $c(S) = \sum_{e \in S} c(e)$. Với một ngưỡng $T > 0$, mục tiêu của bài toán là tìm tập $S \subseteq V$ sao cho: $f(S) \geq T \quad$ và $\quad c(S)$ là nhỏ nhất. Hoặc bài toán được phát biểu như sau:

$$\arg \min_{S \subseteq V : f(S) \geq T} c(S).$$

Một bài toán thuộc lớp MSC có thể được biểu diễn dưới dạng bộ ba (V, f, T). Bổ sung vào bộ ba này một hàm chi phí $c(S) = \sum_{e \in S} c(e)$, các đại lượng liên quan được định nghĩa như sau: $c_{\min} = \min_{e \in V} c(e)$, $c_{\max} = \max_{e \in V} c(e)$, $\rho = \frac{c_{\max}}{c_{\min}}$ và O là nghiệm tối ưu với chi phí tối ưu $\mathsf{opt} = c(O)$. Khác với bài toán SMK (trong chương 2), trong bài toán MSC hàm mục tiêu của bài toán là hàm chi phí $c(\cdot)$, khi đó hàm submodular $f(\cdot)$ trở thành ràng buộc của bài toán.

Bài toán MSC có nhiều ứng dụng thực tiễn trong các hệ thống cần đạt một mức hiệu quả nhất định nhưng với chi phí triển khai tối thiểu. Trong lĩnh vực tiếp thị trên mạng xã hội, MSC được sử dụng để chọn ra tập nhỏ nhất những người ảnh hưởng sao cho thông tin lan tỏa đến một tỷ lệ người dùng mục tiêu, từ đó tối ưu hóa chi phí quảng bá. Trong hệ thống giám sát môi trường hoặc mạng cảm biến không dây, bài toán đặt ra là lựa chọn vị trí lắp đặt tối thiểu số cảm biến sao cho vùng phủ thông tin đạt mức yêu cầu, qua đó tiết kiệm ngân sách đầu tư thiết bị. Trong hệ thống khuyến nghị, MSC hỗ trợ lựa chọn một tập đại diện sản phẩm hoặc nội dung sao cho đảm bảo độ phủ thông tin hoặc đa dạng chủ đề nhưng vẫn giới hạn chi phí gợi ý hoặc hiển thị. Ngoài ra, bài toán cũng được ứng dụng trong việc chọn đặc trưng (feature selection) trong học máy, nơi mục tiêu là chọn ra tập nhỏ nhất các đặc trưng để đạt được một ngưỡng độ chính xác nhất định trong mô hình, đồng thời giảm thiểu chi phí tính toán hoặc thu thập dữ liệu. Những ứng dụng này cho thấy bài toán MSC phù hợp với các hệ thống yêu cầu hiệu suất "đủ tốt" trong điều kiện tài nguyên hạn chế, thay vì tối đa toàn phần.

4.1.2. Nghiên cứu liên quan

Bài toán MSC đã được chứng minh là NP-khó và không thể xấp xỉ trong bất kỳ tỉ lệ nào nhỏ hơn $1 - o(1)$ trừ khi $NP \subset DTIME(n^{O(\log \log n)})$ [42], trong đó $DTIME$ là lớp các bài toán có thể giải trong thời gian đơn định siêu đa thức[1]. Thuật toán tham lam là một trong những cách tiếp cận sớm và tiềm năng để giải bài toán này. Phiên bản tham lam đầu tiên được Wolsey [125] đề xuất vào năm 1982. Thuật toán của họ dựa trên hai đại lượng α và $\beta > 0$, lần lượt biểu thị giá trị mục tiêu lớn nhất trên một phần tử và lợi ích biên nhỏ nhất khác không. Thuật toán này đạt được tỉ lệ xấp xỉ $1 + \ln(\frac{\alpha}{\beta})$. Vào năm 2010, Wan và cộng sự [121] đã cải thiện tỉ lệ xấp xỉ lên $1 + \ln(\frac{\alpha}{\beta})$; tuy nhiên, thuật toán này chỉ áp dụng được cho một trường hợp đặc biệt của bài toán. Để nâng cao hiệu quả, một số tác giả đã nghiên cứu tiếp cận bài toán bằng cách xấp xỉ cả hàm mục tiêu và hàm chi phí – gọi là thuật toán xấp xỉ hai tiêu chí *(Định nghĩa 1.7)*. Cụ thể, Amit Goyal và cộng sự [49] đã đề xuất một phiên bản tham lam có bảo đảm xấp xỉ hai tiêu chí là $(\log(\frac{1}{\epsilon}), 1 - \epsilon)$ dưới một điều kiện dừng cụ thể. Ngoài ra, Crawford và các cộng sự [27] cho thấy thuật toán tham lam cũng có thể cung cấp một giới hạn thay thế trong môi trường có nhiễu, khi mà hàm mục tiêu chỉ có thể được ước lượng với một sai số nhất định.

Bên cạnh phương pháp tham lam truyền thống, một số nghiên cứu khác đã giải quyết bài toán MSC bằng các phương pháp thay thế như thuật toán tiến hoá [28] và thuật toán song song [106]. Nhìn chung, các nghiên cứu trước đây chủ yếu dựa vào phương pháp tham lam. Tuy nhiên, phương pháp này gặp phải hạn chế lớn về độ phức tạp truy

[1]Một hàm được gọi là siêu đa thức nếu nó tăng nhanh hơn bất kỳ hàm đa thức nào, tức là không bị chặn trên bởi n^k với mọi hằng số $k > 0$. Ví dụ điển hình là $n^{\log n}$ hoặc $2^{\sqrt{\log n}}$.

vấn, thường là $O(n^2)$ và đòi hỏi phải quét qua tập dữ liệu $O(n)$ lần, làm cho thuật toán không khả thi với các bộ dữ liệu quy mô lớn. Các phương pháp thay thế, bao gồm thuật toán song song và tiến hoá, vẫn có độ phức tạp tính toán cao với số lần quét dữ liệu có bậc đa thức; thêm vào đó, phương pháp của Yingli Ran và các cộng sự [106] chỉ áp dụng cho các hàm mục tiêu nguyên.

Gần đây, mô hình thuật toán luồng *(Định nghĩa 1.11)* đã nổi lên như một cách tiếp cận mới để giải quyết các bài toán tối ưu hoá, bao gồm cả tối đa các hàm submodular. Mô hình này đã được áp dụng để thiết kế các thuật toán cho bài toán tối đa hàm submodular dưới nhiều ràng buộc khác nhau như ràng buộc lực lượng [5, 47, 74, 97], ràng buộc cấu trúc [18, 77], ràng buộc chi phí [60] và thậm chí được mở rộng cho hàm k-submodular [37, 103], ...

Liên quan đến việc áp dụng mô hình thuật toán luồng để giải bài toán MSC. Một trường hợp đặc biệt của bài toán MSC là bài toán phủ submodular (SC) đã được Norouzi-Fard và các cộng sự [97] nghiên cứu trước. Nhóm tác giả này đã cung cấp một thuật toán bảo đảm xấp xỉ hai tiêu chí $\left(2\ln\left(\frac{1}{\epsilon}\right), 1-\frac{1}{\ln\left(\frac{1}{\epsilon}\right)}\right)$, trong khi chỉ sử dụng bộ nhớ giới hạn ở mức M, với M là kích thước bộ nhớ được chỉ định trước. Đặc biệt, nghiên cứu này cũng chỉ ra bất kỳ thuật toán luồng nào chỉ quét qua dữ liệu một lần mà đạt được hệ số xấp xỉ tốt hơn $\frac{n}{2}$ đều phải sử dụng ít nhất $O(n)$ bộ nhớ đối với bài toán MSC. Gần đây hơn, Crawford và cộng sự [26] đã giới thiệu hai thuật toán, SINGLE và MULTI, cho bài toán MSC. Thuật toán MULTI cung cấp tỉ lệ xấp xỉ hai tiêu chí $((1+\epsilon)\left(1+\frac{4}{\epsilon^2}\right), \frac{1-\epsilon}{2})$ với độ phức tạp truy vấn là $O\left(\frac{n}{\epsilon}\log\left(\frac{\mathsf{opt}}{c_{\min}}\right)\right)$ và quét qua luồng dữ liệu $O(\frac{\log(\mathsf{opt})}{c_{\min}})$ lần. Trong khi đó, thuật toán SINGLE chỉ quét qua dữ liệu một lần và đạt cùng tỉ lệ xấp xỉ, nhưng có độ phức tạp truy vấn cao hơn nhiều ở mức $\Omega(\frac{n^2}{c_{\min}})$. Cần nhấn mạnh độ phức tạp của thuật toán MULTI phụ thuộc vào $c_{\min}$, vốn không phải là hằng số và có thể rất lớn khi $c_{\min}$ nhỏ. Chi tiết so sánh các thuật toán cho bài toán MSC có thể xem ở Bảng 4.1.

Mô hình thuật toán luồng thể hiện ưu thế rõ rệt trong việc xử lý dữ liệu quy mô lớn nhờ khả năng chỉ truy cập và xử lý một phần nhỏ của tập dữ liệu tại mỗi thời điểm. Tuy nhiên, đối với bài toán MSC, các thuật toán luồng hiện có như SINGLE và MULTI dù đạt được tỉ lệ xấp xỉ tốt, vẫn tồn tại những hạn chế đáng kể. Cụ thể, SINGLE yêu cầu độ phức tạp truy vấn cao, trong khi độ phức tạp truy vấn của MULTI lại phụ thuộc vào tham số $c_{\min}$, khiến nó kém thuận lợi trong các trường hợp $c_{\min}$ rất nhỏ. Từ thực tế này, một câu hỏi nghiên cứu tự nhiên được đặt ra là: *Liệu có thể xây dựng một thuật toán luồng với tỉ lệ xấp xỉ tiêu chí kép tốt hơn* MULTI *và đồng thời đạt độ phức tạp truy vấn thấp hơn?*

Bảng 4.1: Bảng so sánh các thuật toán cho bài toán MSC

Thuật toán	Lượt quét	Tỉ lệ xấp xỉ	Độ phức tạp truy vấn	Bộ nhớ sử dụng
SingStr	1	$\left(\frac{(1+\epsilon)(2-\epsilon)}{\epsilon}, 1-\epsilon\right)$	$O(\frac{n}{\epsilon}\log(\frac{\rho n}{\epsilon}))$	$O(\frac{k}{\epsilon}\log(\frac{\rho n}{\epsilon}))$
ThreeStr	3	$\left(\frac{(1+\epsilon)(2-\epsilon)}{\epsilon}, 1-\epsilon\right)$	$\boldsymbol{O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))}$	$\boldsymbol{O(\frac{k}{\epsilon}\log(\frac{n}{\epsilon}))}$
MultiStr	$O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$	$\left(\boldsymbol{(1+\ln(\frac{1}{\epsilon}))\frac{1+2\epsilon}{1-\epsilon}, 1-\epsilon}\right)$	$\boldsymbol{O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon}))}$	$O(\log(\frac{1}{\epsilon})\frac{n}{\epsilon^2}\log(\frac{n}{\epsilon}))$
MULTI [26]	$\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\log(1+\epsilon)}$	$\left((1+\epsilon)(\frac{4}{\epsilon^2}+1), 1-\epsilon\right)$	$O\left(\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\epsilon\log(1+\epsilon)}(n+\frac{\mathsf{opt}}{\epsilon c_{min}})\right)$	$(1+\epsilon)(\frac{4}{\epsilon^2}+1)\mathsf{opt}$
SINGLE [26]	1	$\left((1+\epsilon)(\frac{4}{\epsilon^2}+1), 1-\epsilon\right)$	$O\left(\frac{nB\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\epsilon^2 c_{min}\log(1+\epsilon)}\right)$	$\frac{B(1+\epsilon)(\frac{4}{\epsilon^2}+1)\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\log(1+\epsilon)}$
Greedy [49]	$O(n)$	$\left(1+\ln(\frac{1}{\epsilon}), 1-\epsilon\right)$	$O(n^2)$	$O(k)$

Các giá trị in đậm thể hiện kết quả tốt nhất trong từng tiêu chí. Bộ đôi (α, β) trong cột "Tỉ lệ xấp xỉ" tương ứng với thuật toán xấp xỉ hai tiêu chí (α, β). Các thuật toán được đề xuất trong luận án (SingStr, ThreeStr và MultiStr) sử dụng độ phức tạp bộ nhớ để đo lường mức sử dụng bộ nhớ, trong đó k là kích thước lớn nhất của một nghiệm thỏa mãn trong bài toán MSC:

$$k = \max\{|S| : S \subseteq V,\ f(S) \geq T \text{ và } \exists e \in S,\ f(S \setminus \{e\}) < T\}.$$

Các thuật toán MULTI và SINGLE [26] đánh giá bộ nhớ dựa trên tổng chi phí của các nghiệm ứng viên, do đó độ phức tạp bộ nhớ của chúng có thể không phải là một hàm đa thức.

4.2. Thuật toán đề xuất

Phần này, luận án đề xuất ba thuật toán luồng nhằm giải bài toán MSC. Các thuật toán này được thiết kế nhằm đạt được tỷ lệ xấp xỉ hai tiêu chí với các mức đánh đổi khác nhau giữa số lượt quét, độ phức tạp truy vấn và bộ nhớ. Cụ thể:

SingStr là thuật toán một lượt quét đạt tỷ lệ xấp xỉ hai tiêu chí $(\frac{(1+\epsilon)(2-\epsilon)}{\epsilon},\ 1-\epsilon)$, với độ phức tạp truy vấn $O\left(\frac{n}{\epsilon}\log(\rho n)\right)$ và bộ nhớ $O\left(\frac{k}{\epsilon}\log(\rho n)\right)$. Thuật toán phù hợp với bối cảnh chỉ cho phép đọc dữ liệu một lần, chấp nhận đánh đổi bằng việc duy trì nhiều nghiệm ứng viên theo các ước lượng chi phí khác nhau.

ThreeStr mở rộng từ SingStr bằng cách thực hiện ba lượt quét dữ liệu. Hai lượt quét đầu giúp xác định khoảng chi phí cần xét và rút gọn tập phần tử, lượt cuối áp dụng cơ chế chọn phần tử tương tự SingStr trên bài toán đã được điều chỉnh. Thuật toán đạt cùng tỷ lệ xấp xỉ với SingStr, nhưng giảm độ phức tạp truy vấn xuống $O\left(\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$ và bộ nhớ xuống $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$.

MultiStr là thuật toán nhiều lượt quét, đạt tỷ lệ xấp xỉ hai tiêu chí

$$\left(\left(1+\ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon},\ 1-\epsilon\right)$$

với số lượt quét $O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$. Độ phức tạp truy vấn là $O\left(\log\left(\frac{1}{\epsilon}\right)\frac{n}{\epsilon^2}\log\left(\frac{n}{\epsilon}\right)\right)$ và bộ nhớ là $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$. Thuật toán này phù hợp khi có thể quét dữ liệu nhiều lượt để đổi lấy hệ số xấp xỉ gần hơn với phương pháp tham lam.

Việc đề xuất các thuật toán 1, 3 và nhiều lượt quét xuất phát từ đặc thù của mô hình dữ liệu dạng luồng, trong đó số lần truy cập dữ liệu thường bị giới hạn bởi điều

kiện lưu trữ hoặc chi phí tính toán. Do đó, các thuật toán được thiết kế với số lượt quét khác nhau nhằm bảo đảm tính linh hoạt trong các bối cảnh thực tế. Về cấu trúc, chúng không chỉ khác ở số lần quét mà còn ở cơ chế cập nhật và mức độ tinh chỉnh nghiệm qua từng lượt. Khi số lượt quét tăng, thuật toán có thêm thông tin để cải thiện chất lượng lời giải, thể hiện sự đánh đổi giữa chi phí truy cập dữ liệu và chất lượng nghiệm thu được.

Nhằm thuận tiện cho việc trình bày và phân tích các thuật toán, đặc biệt là phần chứng minh bảo đảm lý thuyết, luận án nhắc lại một số ký hiệu sẽ sử dụng trong các mục tiếp theo tại Bảng 4.2.

Bảng 4.2: Bảng các ký tự toán học dùng trong phân tích các thuật toán luồng cho bài toán MSC

Ký hiệu	**Ý nghĩa**
O	Nghiệm tối ưu của bài toán MSC và $\mathsf{opt} = c(O)$
$c_{\min}$	$c_{\min} = \min_{e \in V} c(e)$
$c_{\max}$	$c_{\max} = \max_{e \in V} c(e)$
ρ	$\rho = \frac{c_{\max}}{c_{\min}}$

4.2.1. Thuật toán luồng một lượt quét SingStr

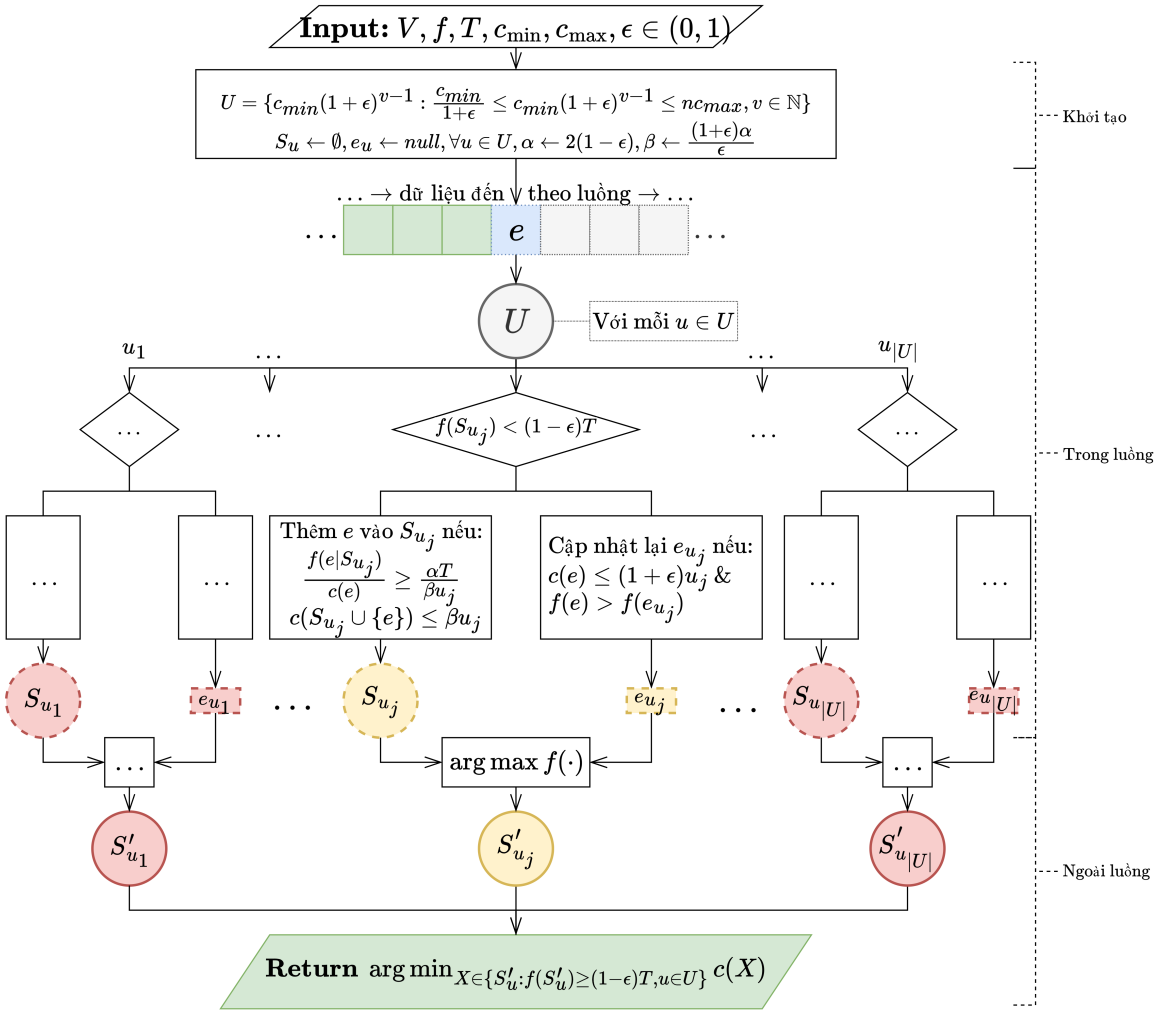
Thuật toán SingStr (Thuật toán 13) được thiết kế để giải bài toán MSC trong mô hình luồng với chỉ một lượt quét qua tập cơ sở V. Thuật toán nhận đầu vào là một bộ (f, V, T), trong đó f là hàm mục tiêu submodular đơn điệu, V là tập cơ sở, T là ngưỡng bao phủ; cùng với đó là các tham số $c_{\min}$, $c_{\max}$ biểu thị chi phí nhỏ nhất và lớn nhất trong V và tham số chính xác $\epsilon > 0$.

Ở bước khởi tạo, thuật toán xác định tập U gồm các ước lượng khả dĩ của nghiệm tối ưu, tương ứng với mỗi $u \in U$ là một nghiệm ứng viên S_u và một phần tử e_u. Hai tham số α và β được thiết lập lần lượt bằng $1 - \epsilon$ và $\beta = \frac{(1+\epsilon)\alpha}{\epsilon}$ nhằm điều chỉnh ngưỡng chọn phần tử (dòng 1–2). Trong quá trình quét duy nhất qua tập V, thuật toán kiểm tra mỗi phần tử $e \in V$ để quyết định xem có nên thêm vào mỗi nghiệm ứng viên S_u hay không. Phần tử e được chọn đưa vào S_u nếu thoả mãn đồng thời hai điều kiện (dòng 6–7):

1) $f(S_u) < T(1 - \epsilon)$
2) $\frac{f(e \mid S_u)}{c(e)} \geq \frac{\alpha T}{\beta u}$ và $c(S_u \cup \{e\}) \leq \beta u$

Điều kiện thứ nhất được sử dụng để kiểm tra ngưỡng bao phủ của nghiệm ứng viên, còn điều kiện thứ hai yêu cầu phần tử e có mật độ lợi ích biên đủ lớn và tổng chi phí không vượt quá βu. Ngoài ra, nếu phần tử e thoả $c(e) \leq (1 + \epsilon)u$, thuật toán cập nhật e_u là phần tử có giá trị lớn nhất thoả mãn điều kiện chi phí (dòng 9–10). Sau khi quét hết tập V, mỗi nghiệm ứng viên S_u được cập nhật bằng cách chọn nghiệm tốt nhất giữa S_u và $\{e_u\}$ (dòng 11). Cuối cùng, thuật toán trả về nghiệm tốt nhất trong số các S'_u ứng

với mọi $u \in U$ (dòng 12–13). Cụ thể, các bước được trình bày trong Thuật toán 13 và được minh họa trong Hình 4.1.



Hình 4.1: Lưu đồ hoạt động của thuật toán một lượt quét SingStr

Tiếp theo, luận án trình bày phân tích lý thuyết và bảo đảm xấp xỉ của thuật toán SingStr thông qua Định lý 4.1.

Phác thảo chứng minh. Ý tưởng chứng minh của thuật toán SingStr là duy trì đồng thời nhiều nghiệm ứng viên ứng với các ước lượng khác nhau của chi phí tối ưu. Vì tập các ước lượng U phủ toàn bộ khoảng chi phí khả dĩ theo cấp số nhân, luôn tồn tại một giá trị u gần với opt. Với giá trị này, chứng minh tách thành hai tình huống: hoặc một phần tử tối ưu không thể được thêm vì ràng buộc chi phí ứng viên đã gần đầy, hoặc mọi phần tử tối ưu còn lại không đạt ngưỡng mật độ lợi ích biên. Trong cả hai tình huống, tính đơn điệu và submodular của f cho phép chặn phần giá trị chưa bao phủ và suy ra nghiệm cuối cùng đạt ngưỡng $(1-\epsilon)T$ với chi phí bị chặn theo opt.

Định lý 4.1. *Thuật toán 13 là một thuật toán luồng một lượt quét, có độ phức tạp truy vấn* $O\left(\frac{n}{\epsilon}\log(\rho n)\right)$*, độ phức tạp bộ nhớ* $O\left(\frac{k}{\epsilon}\log(\rho n)\right)$ *và trả về nghiệm* S *sao cho* $f(S) \ge (1-\epsilon)T$ *và* $c(S) \le \frac{(1+\epsilon)(2-\epsilon)}{\epsilon}$opt.

Algorithm 13 Thuật toán một lượt quét SingStr

Input: An instance (V, f, T), $c_{\min}, c_{\max}$, $\epsilon \in (0, 1)$
Output: A solution S of MSC

$U = \{c_{min}(1+\epsilon)^{v-1} : \frac{c_{min}}{1+\epsilon} \leq c_{min}(1+\epsilon)^{v-1} \leq nc_{max}, v \in \mathbb{N}\}$
$S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
$\alpha \leftarrow 2(1-\epsilon), \beta \leftarrow \frac{(1+\epsilon)\alpha}{\epsilon}$
foreach *incoming element* $e \in V$ **do**
    foreach $u \in U$ **do**
        if $f(S_u) < (1-\epsilon)T$ **then**
            if $\frac{f(e|S_u)}{c(e)} \geq \frac{\alpha T}{\beta u}$ *and* $c(S_u \cup \{e\}) \leq \beta u$ **then**
                $S_u \leftarrow S_u \cup \{e\}$
            if $c(e) \leq (1+\epsilon)u$ **then**
                $e_u \leftarrow \arg\max_{x \in \{e, e_u\}} f(x)$
$S'_u \leftarrow \arg\max_{X \in \{\{e_u\}, S_u\}} f(X), \forall u \in U$
$S \leftarrow \arg\min_{X \in \{S'_u : f(S'_u) \geq (1-\epsilon)T, u \in U\}} c(X)$
return S

Chứng minh. **Phân tích tỷ lệ xấp xỉ.** Do $c_{\min} \leq \mathsf{opt} \leq c(V)$, tồn tại một số nguyên v và một giá trị $u \in U$ sao cho:

$$\frac{\mathsf{opt}}{1+\epsilon} \leq u = c_{\min}(1+\epsilon)^{v-1} \leq \mathsf{opt}.$$

Với giá trị u như vậy, ta phân tích nghiệm ứng viên S'_u và xét hai trường hợp sau:

Trường hợp 1. Tồn tại phần tử $o \in O \setminus S_u$ sao cho $c(S_u)+c(o) > \beta u$ và $\frac{f(o \mid S_u)}{c(o)} \geq \frac{\alpha T}{\beta u}$. Theo quy tắc chọn phần tử vào S_u, ta có: $f(S_u) \geq \frac{c(S_u)\alpha T}{\beta u}$. Do tính submodular của f:

$$\begin{aligned} f(S_u \cup \{o\}) &= f(S_u) + f(o \mid S_u) \\ &\geq \frac{c(S_u)\alpha T}{\beta u} + \frac{\alpha c(o) T}{\beta u} \\ &= (c(S_u) + c(o))\frac{\alpha T}{\beta u} \\ &> \alpha T. \end{aligned}$$

Theo cách cập nhật e_u, vì $c(e_u) \leq (1+\epsilon)u > \mathsf{opt}$, suy ra $f(e_u) \geq \max_{o \in O} f(o)$. Kết hợp với tính submodular:

$$f(S'_u) \geq \max\{f(S_u), f(e_u)\} \geq \max\{f(S_u), f(o)\} \geq \frac{f(S_u \cup \{o\})}{2} > \frac{\alpha T}{2}.$$

Đồng thời, ta có:

$$c(S'_u) \leq \max\{c(S_u), c(e_u)\} = \max\{\beta u, (1+\epsilon)u\} \leq \beta u \leq \beta \mathsf{opt}.$$

Trường hợp 2. Không tồn tại phần tử nào thoả điều kiện trên, tức là với mọi $o \in O \setminus S_u$, ta có $c(S_u) + c(o) \leq \beta u$ và $\frac{f(o \mid S_u)}{c(o)} \geq \frac{\alpha T}{\beta u}$. Khi đó: $c(S_u) \leq \beta \mathsf{opt}$. Sử dụng tính

đơn điệu và submodular của f, đồng thời với $u \geq \frac{\mathsf{opt}}{1+\epsilon}$, ta có:

$$\begin{aligned} f(O) - f(S_u) &\leq f(O \cup S_u) - f(S_u) \\ &\leq \sum_{o \in O \setminus S_u} f(o \,|\, S_u) \\ &< \sum_{o \in O \setminus S_u} c(o) \frac{\alpha T}{\beta u} \leq \mathsf{opt} \frac{\alpha T}{\beta u} \\ &\leq \frac{\alpha(1+\epsilon)T}{\beta}. \end{aligned}$$

Do đó: $f(S_u) \geq f(O) - \frac{\alpha(1+\epsilon)T}{\beta} \geq \left(1 - \frac{(1+\epsilon)\alpha}{\beta}\right) T \geq (1-\epsilon)T$. Suy ra $f(S'_u) \geq (1-\epsilon)T$ và $c(S'_u) \leq \frac{(1+\epsilon)(2-\epsilon)}{\epsilon}\mathsf{opt}$.

Theo quy tắc chọn nghiệm cuối cùng, thuật toán chọn S sao cho $f(S) \geq (1-\epsilon)T$ và $c(S) \leq c(S'_u)$, do đó bảo đảm xấp xỉ như đã nêu.

Phân tích độ phức tạp. Thuật toán quét qua tập V đúng một lần. Có $|U|$ nghiệm ứng viên S_u. Mỗi S_u và e_u được cập nhật tối đa $O(n)$ truy vấn, nên tổng số truy vấn là:

$$\begin{aligned} 2n|U| &= 2n \log_{1+\epsilon}(\frac{c(V)}{c_{min}}) \leq 2n \log_{1+\epsilon}(\frac{nc_{max}}{c_{min}}) \\ &= 2n \frac{\ln(\frac{nc_{max}}{c_{min}})}{\log(1+\epsilon)} \leq 2n \frac{\ln(\frac{nc_{max}}{c_{min}})}{\log(\frac{1}{1-\frac{\epsilon}{2}})} \\ &\leq \frac{4n}{\epsilon} \ln(\frac{nc_{max}}{c_{min}}) = O(\frac{n}{\epsilon} \log(\rho n)). \end{aligned}$$

Cuối cùng, theo điều kiện tại dòng 6 của thuật toán, kích thước mỗi S_u là tối đa k, nên tổng bộ nhớ là $|U|(k+1) = O\left(\frac{k}{\epsilon} \log(\rho n)\right)$. □

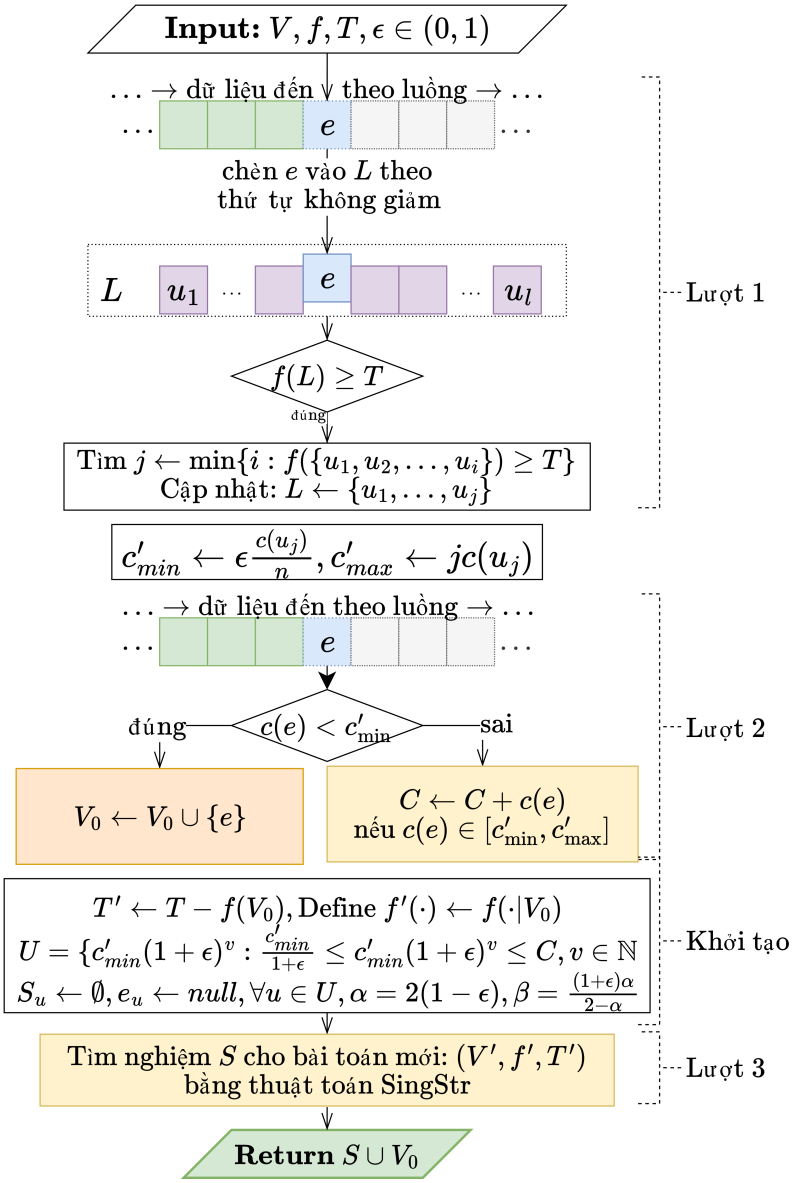
4.2.2. Thuật toán luồng ba lượt quét ThreeStr

Thuật toán ThreeStr (Thuật toán 14) được phát triển nhằm cải thiện hiệu năng của SingStr bằng cách giảm đáng kể độ phức tạp truy vấn trong khi vẫn giữ nguyên tỷ lệ xấp xỉ. Cụ thể, ThreeStr chỉ yêu cầu $O(n \log n)$ truy vấn trong khi đảm bảo tỷ lệ xấp xỉ hai tiêu chí tương tự SingStr, tức là $\left(\frac{(1+\epsilon)(2-\epsilon)}{\epsilon},\ 1-\epsilon\right)$. Thuật toán thực hiện ba lượt quét dữ liệu và sử dụng khung thuật toán mới gồm hai thành phần chính: (1) chia tập cơ sở V thành các tập con hợp lý và (2) điều chỉnh các bước của SingStr để xây dựng nghiệm chất lượng cao với chi phí truy vấn thấp hơn.

Lượt quét thứ nhất (Dòng 1–7) nhằm xây dựng tập $L = \{u_1, u_2, \ldots, u_j\} \subseteq V$ gồm các phần tử có chi phí tăng dần sao cho $f(L) \geq T$. Thuật toán khởi tạo tập L rỗng, sau đó thêm từng phần tử e vào L theo thứ tự tăng dần chi phí và dừng khi điều kiện $f(L) \geq T$ được thoả mãn. Giai đoạn này chỉ yêu cầu bộ nhớ tối đa k đơn vị để lưu trữ tập L. Sau khi kết thúc, hai ngưỡng chi phí mới được xác định: $c'_{\min} = \epsilon \frac{c(u_j)}{n}$ và $c'_{\max} = j \cdot c(u_j)$, dùng trong các bước tiếp theo.

Lượt quét thứ hai (Dòng 8–17) xác định hai tập con $V_0 = \{e \in V : c(e) < c'_{\min}\}$ và $V' = \{e \in V : c'_{\min} \leq c(e) \leq c'_{\max}\}$. Việc xác định rõ ranh giới chi phí này giúp giảm kích thước không gian tìm kiếm lời giải ở lượt quét thứ ba.

Lượt quét cuối cùng (Dòng 18–25) là quá trình tìm nghiệm cho bài toán mới (V', f', T'), trong đó $T' = T - f(V_0)$ và $f'(\cdot) = f(\cdot \mid V_0)$. Thuật toán khởi tạo các nghiệm ứng viên S_u, phần tử e_u với mỗi $u \in U$ cùng với các tham số ngưỡng α và β. Sau đó, ThreeStr điều chỉnh thao tác của SingStr để xây dựng các nghiệm S'_u dựa trên V' và cuối cùng chọn nghiệm tốt nhất từ $\{S'_u\}$ (Dòng 27). Nghiệm cuối cùng được trả về là $S \cup V_0$. Cụ thể, quy trình của thuật toán được nêu trong Thuật toán 14 và minh họa ở Hình 4.2.



Hình 4.2: Lưu đồ hoạt động của thuật toán ba lượt quét ThreeStr

Cách tiếp cận này cho phép ThreeStr kế thừa bảo đảm lý thuyết của SingStr nhưng giảm đáng kể số lượng truy vấn. Ngoài ra, do kích thước của V' được giới hạn bởi L nên bộ nhớ và chi phí tính toán thực tế của ThreeStr cũng thấp hơn so với SingStr. So sánh

với thuật toán SINGLE [26], ThreeStr duy trì tỷ lệ xấp xỉ tương tự nhưng chỉ yêu cầu $O(n \log n)$ truy vấn thay vì $\Omega(n^2)$ và không phụ thuộc vào các tham số bất lợi như $c_{\min}$.

Algorithm 14 Thuật toán ba lượt quét ThreeStr

Input: An instance (V, f, T), $\epsilon \in (0, 1)$
Output: A solution S of MSC

foreach *incoming element* e **do**
  Insert e into L in non-decreasing cost order
  Let $L = \{u_1, u_2, \ldots, u_l\}$
  if $f(L) \geq T$ **then**
    Find $j \leftarrow \min\{i : f(\{u_1, u_2, \ldots, u_i\}) \geq T\}$
    Update: $L \leftarrow \{u_1, \ldots, u_j\}$
$c'_{min} \leftarrow \epsilon \frac{c(u_j)}{n}$, $c'_{max} \leftarrow jc(u_j)$
foreach *incoming element* $e \in V$ **do**
  if $c(e) < c'_{min}$ **then**
    $V_0 \leftarrow V_0 \cup \{e\}$
  else
    if $c(e) \in [c'_{min}, c'_{max}]$ **then**
      $C \leftarrow C + c(e)$
$T' \leftarrow T - f(V_0)$, Define $f'(\cdot) \leftarrow f(\cdot|V_0)$
$U = \{c'_{min}(1+\epsilon)^v : \frac{c'_{min}}{1+\epsilon} \leq c'_{min}(1+\epsilon)^v \leq C, v \in \mathbb{N}\}$
$S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
$\alpha = 2(1-\epsilon)$,$\beta = \frac{(1+\epsilon)\alpha}{2-\alpha}$
foreach *incoming element* $e \in V$ **do**
  if $c(e) \in [c'_{min}, c'_{max}]$ **then**
    foreach $u \in U$ **do**
      if $f'(S_u) < (1-\epsilon)T'$ **then**
        if $\frac{f'(e|S_u)}{c(e)} \geq \frac{\alpha T'}{\beta u}$ *and* $c(S_u \cup \{e\}) \leq \beta u$ **then**
          $S_u \leftarrow S_u \cup \{e\}$
        if $c(e) \leq (1+\epsilon)u$ **then**
          $e_u \leftarrow \arg\max_{x \in \{e, e_u\}} f'(x)$
$S'_u \leftarrow \arg\max_{X \in \{\{e_u\}, S_u\}} f'(X), \forall u \in U$
$S \leftarrow \arg\min_{X \in \{S'_u : f'(S'_u) \geq (1-\epsilon)T', u \in U\}} c(X)$
return $S \cup V_0$.

Phân tích sau đây chứng minh các đảm bảo lý thuyết của thuật toán ThreeStr, bao gồm giới hạn về chi phí nghiệm tối ưu, tỷ lệ xấp xỉ, độ phức tạp truy vấn và độ phức tạp bộ nhớ.

Phác thảo chứng minh. Phân tích của ThreeStr dựa trên việc dùng hai lượt quét đầu để thu hẹp không gian tìm kiếm mà không làm mất nghiệm tối ưu quan trọng. Lượt thứ nhất tạo danh sách L nhằm xác định một khoảng chi phí chứa nghiệm tối ưu; lượt thứ hai tách các phần tử rất nhỏ vào V_0 và chỉ giữ lại các phần tử có chi phí nằm trong

khoảng cần xét. Sau bước rút gọn này, bài toán còn lại trên V' có thể được xử lý tương tự SingStr, nhưng số lượng ước lượng ứng viên và chi phí truy vấn được kiểm soát tốt hơn. Do V_0 có tổng chi phí nhỏ và nghiệm tối ưu vẫn có đại diện trong $V_0 \cup V'$, nghiệm cuối cùng giữ được bảo đảm bao phủ và chi phí.

Bổ đề 4.1. *Sau khi kết thúc lượt quét đầu tiên của Thuật toán 14, danh sách* $L = \{u_1, u_2, \ldots, u_j\}$ *thoả mãn:*

- $c(u_j) \le c(O) \le c(\{u_1, \ldots, u_j\}) \le j \cdot c(u_j)$,
- $|L| \le k$,
- $\mathsf{opt} \le c_{\max}$.

Chứng minh. Dễ dàng thấy L chứa tất cả các phần tử trong V có chi phí không vượt quá $c(v_j)$, tức là $L = \{e \in V : c(e) \le c(v_j)\}$. Giả sử $V = \{v_1, v_2, \ldots, v_n\}$ được sắp xếp theo thứ tự chi phí không giảm và $v_t = \max_{e \in O} c(e)$ với một số $t \le n$. Nếu $c(v_t) < c(v_j)$, khi đó $O = \{v_1, v_2, \ldots, v_t\} \subset L$. Theo tính đơn điệu của f, ta có $f(O) = f(\{v_1, v_2, \ldots, v_t\}) \ge T$, điều này mâu thuẫn với quy tắc lựa chọn của j và thực tế $L = \{e \in V : c(e) \le c(v_j)\}$. Do đó, $c(v_j) \le c(v_t) \le c(O)$. Mặt khác, vì tập $\{v_1, v_2, \ldots, v_j\}$ là một nghiệm khả thi của bài toán (V, f, T), ta có:

$$c(v_j) \le c(O) \le c(\{v_1, v_2, \ldots, v_j\}) \le jc(v_j) = c_{max}.$$

Theo quy tắc lựa chọn của L, nếu loại bỏ v_j khỏi L thì tập này sẽ không còn là nghiệm khả thi nữa, do đó $|L| \le k$. □

Định lý 4.2. *Thuật toán 14 thực hiện ba lượt quét trên tập cơ sở, có độ phức tạp bộ nhớ* $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$*, độ phức tạp truy vấn* $O\left(\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$ *và trả về nghiệm* S *sao cho* $f(S) \ge (1-\epsilon)T$ *và* $c(S) \le \frac{\mathsf{opt}}{\epsilon}$.

Chứng minh. Theo cách lựa chọn của V_0, V' và Bổ đề 4.1, ta có

$$c(V_0) \le |V_0| c'_{min} \le n\frac{\epsilon c(u_j)}{n} \le \epsilon c(v_j) \le \epsilon\mathsf{opt}$$

và $O \subseteq V_0 \cup V'$. Ký hiệu O' là một nghiệm tối ưu của bài toán (V', f', T') và $\mathsf{opt}' = c(O')$, trong đó $f'(\cdot) = f(\cdot|V_0) = f(V_0 \cup \cdot) - f(V_0) \ge T'$ và $T' = T - f(V_0)$. Dễ dàng thấy $f'(\cdot)$ là một hàm đơn điệu và submodular. Vì

$$\begin{aligned} f'(O \cap V') &= f((O \cap V') \cup V_0) - f(V_0) \\ &= f(O \cup V_0) - f(V_0) \ge T - f(V_0) \end{aligned}$$

nên $O \cap V'$ là một nghiệm khả thi của bài toán ứng với (V', f', T'). Do đó $\mathsf{opt}' \le c(O \cap V') \le \mathsf{opt}$.

Hơn nữa, vì $c'_{min} \le u = (1+\epsilon)^v \le c(V')$, tồn tại một số nguyên v sao cho $\frac{\mathsf{opt}'}{1+\epsilon} < u = (1+\epsilon)^v \le \mathsf{opt}'$. Xác định cận lý thuyết của S'_v bằng cách xét hai trường hợp sau:

Trường hợp 1. Tồn tại một phần tử $o \in O' \setminus S_u$ sao cho $c(S_u)+c(o) > \beta u$ và $\frac{f'(o|S_u)}{c(o)} \geq \frac{\alpha T'}{\beta u}$. Theo quy tắc lựa chọn của S_u, ta có $f'(S_u) \geq \frac{c(S_u)\alpha T'}{\beta u}$. Do đó:

$$f'(S_u \cup \{o\}) \geq f'(S_u) + \frac{\alpha c(o) T'}{\beta u} \geq \frac{c(S_u)\alpha T'}{\beta u} + \frac{\alpha c(o) T'}{\beta u} = \Big(c(S) + c(o)\Big)\frac{\alpha T'}{\beta u} > \alpha T'.$$

Theo quy tắc lựa chọn của e_u, ta có $c(e_u) \leq (1+\epsilon)u > \mathsf{opt}'$, nên $f(e_u) \geq \max_{o \in O'} f'(o)$. Kết hợp với tính submodular của f', ta có:

$$f'(S'_u) \geq \max\{f'(S_u), f(e_u)\} \geq \max\{f'(S_u), f'(o)\} \geq \frac{f'(S_u \cup \{o\})}{2} > \frac{\alpha T'}{2}$$

và $c(S'_u) \leq \max\{c(S_u), c(e_u)\} = \max\{\beta u, (1+\epsilon)u\} \leq \beta u \leq \beta \mathsf{opt}'$.

Trường hợp 2. Không tồn tại phần tử như vậy $o \in O' \setminus S_u$, tức là $c(S_u) + c(o) \leq \beta u$ và $\frac{f'(o|S_u)}{c(o)} \geq \frac{\alpha T'}{\beta u}$ với mọi $o \in O' \setminus S_u$. Theo quy tắc lựa chọn của S_u và e_u, ta cũng có:

$$c(S'_u) \leq \max\{c(S_u), c(e_u)\} = \max\{\beta u, (1+\epsilon)u\} \leq \beta u \leq \beta \mathsf{opt}'.$$

Theo tính đơn điệu và submodular của f' với lưu ý $u = (1+\epsilon)^v > \frac{\mathsf{opt}'}{1+\epsilon}$, ta có:

$$\begin{aligned} f'(O') - f'(S_u) &\leq f'(O' \cup S_u) - f'(S_u) \\ &\leq \sum_{o \in O' \setminus S_u} f'(o|S_u) \leq \sum_{o \in O' \setminus S_u} c(o)\frac{\alpha T'}{\beta(1+\epsilon)^u} \\ &\leq \mathsf{opt}'\frac{\alpha T'}{\beta u} < \frac{\alpha(1+\epsilon)T'}{\beta} \end{aligned}$$

suy ra:

$$f'(S_u) \geq f'(O') - \frac{\alpha(1+\epsilon)T'}{\beta} \geq T' - \frac{\alpha(1+\epsilon)T'}{\beta} = \Big(1 - \frac{(1+\epsilon)\alpha}{\beta}\Big)T'.$$

Kết hợp hai trường hợp với cách thiết lập α, β trong thuật toán, ta có $f'(S'_u) \geq (1-\epsilon)T'$ và $c(S'_u) \leq \frac{1-\epsilon^2}{\epsilon}\mathsf{opt}'$. Do đó, thuật toán luôn tìm được nghiệm cuối cùng S với một số $u \in U$ tại Dòng 27 sao cho $f'(S) \geq (1-\epsilon)T'$ và $c(S) \leq c(S'_u)$. Nhắc lại $c(V_0) \leq \epsilon\mathsf{opt}$ và $\mathsf{opt}' \leq \mathsf{opt}$, ta có:

$$c(S \cup V_0) \leq c(S'_u \cup V_0) \leq (\beta + \epsilon)\mathsf{opt} \leq \frac{\mathsf{opt}}{\epsilon}.$$

Mặt khác,

$$\begin{aligned} f(S \cup V_0) &= f(S \cup V_0) - f(V_0) + f(V_0) \\ &= f'(S) + f(V_0) \geq (1-\epsilon)T' + f(V_0) \\ &= (1-\epsilon)(T - f(V_0)) + f(V_0) \\ &= (1-\epsilon)T + \epsilon f(V_0) \geq (1-\epsilon)T. \end{aligned}$$

Như vậy, bảo đảm xấp xỉ được chứng minh.

Lượt quét thứ nhất cần $O(n)$ truy vấn. Lượt quét thứ hai cần $O(n)$ truy vấn để tìm V_0 và C. Trong lượt quét thứ ba, thuật toán xử lý mỗi phần tử đến trong luồng với nhiều

nhất

$$\begin{aligned}|U| &\le \log_{1+\epsilon}(\frac{C}{c'_0}) + 1 \le \log_{1+\epsilon}(n\frac{c'_{max}}{c'_{min}}) + 1 \\ &\le \log_{1+\epsilon}(n\frac{n\cdot j}{\epsilon}) + 1 \le \log_{1+\epsilon}(\frac{n^3}{\epsilon}) + 1 \\ &= \frac{\log(\frac{n^3}{\epsilon})}{\log(1+\epsilon)} + 1 \le \frac{1}{\epsilon}\log(\frac{n^3}{\epsilon}) + 1 = O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})).\end{aligned}$$

Do đó độ phức tạp truy vấn là

$$O(n) + O(n) + O(\frac{n}{\epsilon}\log(\frac{n}{\epsilon})) = O\Big(\frac{n}{\epsilon}\log(\frac{n}{\epsilon})\Big).$$

Lượt quét đầu tiên cần k bộ nhớ để lưu L. Lượt quét thứ hai cần $|V_0| < k$ bộ nhớ vì V_0 không phải là nghiệm khả thi. Trong lượt quét cuối, thuật toán sử dụng $|S_u|$ bộ nhớ để lưu S_u. Với mỗi $u \in U$, thuật toán cập nhật S_u cho đến khi đạt điều kiện $f'(S_u) \ge (1-\epsilon)T'$ hoặc khi kết thúc lượt quét cuối. Do đó tồn tại một phần tử $e \in S_u$ sao cho $f'(S_u \setminus \{e\}) < (1-\epsilon)T'$. Suy ra:

$$\begin{aligned}f(S_u \setminus \{e\}) &\le f((S_u \setminus \{e\}) \cup V_0) \\ &= f(S_u \setminus \{e\} \cup V_0) - f(V_0) + f(V_0) \\ &= f'(S_u \setminus \{e\}) + f(V_0) < (1-\epsilon)T' + f(V_0) \\ &= T - \epsilon T' < T\end{aligned}$$

từ đó suy ra $|S_u| \le k$ với mọi $u \in U$. Do đó, mức sử dụng bộ nhớ trong lượt quét cuối nhiều nhất là $k|U|$ và độ phức tạp bộ nhớ là

$$k + k + k|U| = k + k + O(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})) = O\Big(\frac{k}{\epsilon}\log(\frac{n}{\epsilon})\Big).$$

Chứng minh được hoàn thành. □

4.2.3. Thuật toán luồng nhiều lượt quét MultiStr

MultiStr (Thuật toán 15), được thiết kế để cải thiện đáng kể tỷ lệ xấp xỉ hai tiêu chí trong khi vẫn duy trì độ phức tạp truy vấn tương đương với thuật toán ThreeStr. Cụ thể, MultiStr thực hiện $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$ lượt quét qua toàn bộ dữ liệu, giữ nguyên độ phức tạp truy vấn của ThreeStr là $O\left(\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$, nhưng đạt được tỷ lệ xấp xỉ cải tiến $\left(\left(1+\ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon},\ 1-\epsilon\right)$.

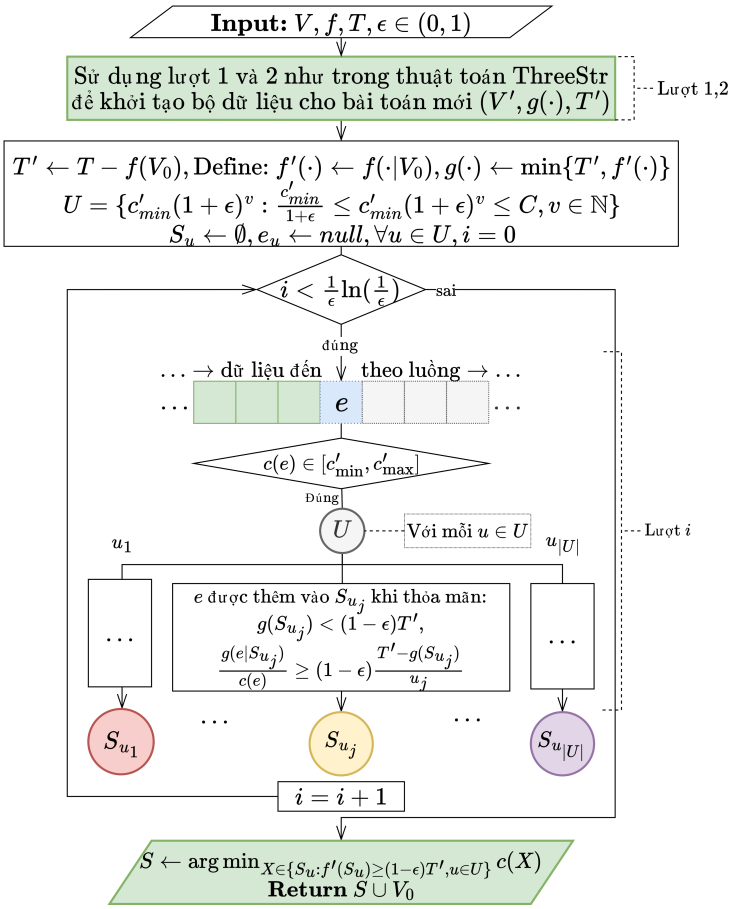
Ý tưởng chính để nâng cao tỷ lệ xấp xỉ là cập nhật các nghiệm ứng viên bằng cách điều chỉnh ngưỡng tham lam, theo hướng tiếp cận của Badanidiyuru và cộng sự [6], nhằm lọc ra các phần tử có mật độ lợi ích biên cao phù hợp với trạng thái hiện tại của nghiệm. Hai lượt quét đầu tiên của thuật toán MultiStr tương tự như ThreeStr: lượt đầu nhằm xác định tập L và các ngưỡng chi phí $c'_{\min}, c'_{\max}$; lượt thứ hai xác định tập V_0 và xác lập phiên bản hàm mục tiêu mới $f'(\cdot)$ cùng ngưỡng mục tiêu $T' = T - f(V_0)$. Sau hai lượt quét đầu, thuật toán tiến hành một vòng lặp chính gồm $\log\left(\frac{1}{\epsilon}\right)$ lần lặp. Gọi

$g = \min\{T', f'(\cdot)\}$. Trong mỗi lần lặp, các nghiệm ứng viên S_u, với $u \in U$, được cập nhật bằng cách thêm các phần tử e nếu đồng thời thỏa mãn hai điều kiện sau:

1) $g(S_u) \leq T'(1-\epsilon)$,

2) $\frac{g(e \mid S_u)}{c(e)} \geq (1-\epsilon) \cdot \frac{T'-g(S_u)}{u}$.

Sự khác biệt then chốt giữa ThreeStr và MultiStr nằm ở điều kiện thứ hai trong Dòng 22 của Thuật toán 15. Điều kiện này cho phép thêm các phần tử có mật độ lợi ích biên khác nhau vào các lời giải ứng viên. Nhờ đó, thuật toán có thể nâng cao chất lượng nghiệm mà vẫn đảm bảo tổng chi phí không vượt ngưỡng đã định. Cụ thể, các bước được trình bày trong Thuật toán 15 và được minh họa trong Hình 4.3.



Hình 4.3: Lưu đồ hoạt động của thuật toán nhiều lượt quét MultiStr

Tiếp theo, Luận án trình bày kết quả lý thuyết cho thuật toán MultiStr trong hai bước: (i) thiết lập giới hạn trên cho chi phí và bảo đảm xấp xỉ thông qua Bổ đề 4.2 và (ii) tổng hợp kết quả về độ chính xác và độ phức tạp trong Định lý 4.3.

Phác thảo chứng minh. Thuật toán MultiStr kế thừa hai lượt tiền xử lý của ThreeStr, sau đó cải thiện chất lượng nghiệm bằng nhiều lượt cập nhật với ngưỡng phụ thuộc vào phần giá trị còn thiếu so với T'. Mỗi lượt quét làm giảm phần chưa bao phủ theo một tỉ lệ nhân, nên sau $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ lượt, nghiệm đạt ít nhất $(1-\epsilon)T'$. Đồng thời, tổng chi phí các phần tử được thêm vào được chặn bằng cách cộng dồn các mức giảm giá trị

Algorithm 15 Thuật toán luồng nhiều lượt quét MultiStr

Input: An instance (V, f, T), $\epsilon \in (0, 1)$
Output: A solution S of MSC

foreach *incoming element* e **do**
  Insert e into L in non-decreasing cost order
  Let $L = \{u_1, u_2, \ldots, u_l\}$
  if $f(L) \geq T$ **then**
    Find $j \leftarrow \min\{i : f(\{u_1, u_2, \ldots, u_i\}) \geq T\}$
    Update: $L \leftarrow \{u_1, \ldots, u_j\}$
$c'_{min} \leftarrow \epsilon\frac{c(u_j)}{n}$, $c'_{max} \leftarrow jc(u_j)$
foreach *incoming element* $e \in V$ **do**
  if $c(e) < c'_{min}$ **then**
    $V_0 \leftarrow V_0 \cup \{e\}$
  else
    if $c(e) \in [c'_{min}, c'_{max}]$ **then**
      $C \leftarrow C + c(e)$
$T' \leftarrow T - f(V_0)$, define $f'(\cdot) \leftarrow f(\cdot|V_0)$
$U = \{c'_{min}(1+\epsilon)^v : \frac{c'_{min}}{1+\epsilon} \leq c'_{min}(1+\epsilon)^v \leq C, v \in \mathbb{N}\}$ $S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
Define $g(\cdot) \leftarrow \min\{T', f'(\cdot)\}$
for $i = 1$ *to* $\frac{1}{\epsilon}\ln(\frac{1}{\epsilon})$ **do**
  foreach *incoming element* $e \in V$ **do**
    if $c(e) \in [c'_{min}, c'_{max}]$ **then**
      foreach $u \in U$ **do**
        if $g(S_u) < (1-\epsilon)T'$ **then**
          if $\frac{g(e|S_u)}{c(e)} \geq (1-\epsilon)\frac{T'-g(S_u)}{u}$ **then**
            $S_u \leftarrow S_u \cup \{e\}$
$S \leftarrow \arg\min_{X \in \{S_u : f'(S_u) \geq (1-\epsilon)T', u \in U\}} c(X)$
return $S \cup V_0$.

qua từng lượt, tạo ra hệ số dạng logarit gần với phân tích của thuật toán tham lam. Kết hợp với cận chi phí của V_0 cho ra bảo đảm cuối cùng cho nghiệm $S \cup V_0$.

Bổ đề 4.2. *Với thể hiện (V', g, T') của bài toán MSC sau dòng 16 của Thuật toán 15, trong đó $T' := T - f(V_0)$, $g(\cdot) := \min\{f'(\cdot), T'\}$ và $V' = \{e \in V : c'_{\min} \leq c(e) \leq c'_{\max}\}$, giả sử tồn tại $u \in U$ sao cho $\frac{u}{1+\epsilon} \leq \mathsf{opt}' \leq u$. Giả sử thuật toán thu được S_u sau dòng 23, thì ta có: $f'(S_u) \geq T'(1-\epsilon)$ và $c(S_u) \leq \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+\epsilon}{1-\epsilon}\mathsf{opt}'$.*

Chứng minh. Ký hiệu S_u^i là S_u sau khi kết thúc vòng lặp thứ $i, 1 \leq i \leq l = \frac{1}{\epsilon}\ln(\frac{1}{\epsilon})$. Vì f' là một hàm đơn điệu và submodular, nên $g = \min\{T', f'(\cdot)\}$ cũng là một hàm đơn điệu và submodular. Mỗi phần tử không được thêm vào tại vòng lặp i có mật độ lợi ích biên nhỏ hơn $(1-\epsilon)\frac{T'-f'(S_u^i)}{u}$, do đó:

$$T' - g(S_u^i) \leq g(O) - g(S_u^i)$$

$$
\begin{aligned}
&\leq g(O \cup S_u^i) - g(S_u^i) \quad \text{(theo tính đơn điệu của } g) \\
&\leq \sum_{e \in O \setminus S_u^i} g(e|S_u^i) \quad \text{(theo tính submodular của } g) \\
&\leq \sum_{e \in O \setminus S_u^i} c(e)(1-\epsilon)\frac{T' - g(S_u^i)}{u} \\
&\leq \mathsf{opt}'(1-\epsilon)\frac{T' - g(S_u^i)}{u} \\
&\leq (1-\epsilon)(T' - g(S_u^i))
\end{aligned}
$$

suy ra:

$$T' - g(S_u^l) \leq (1-\epsilon)^l T' \leq e^{-\epsilon\frac{1}{\epsilon}\ln(\frac{1}{\epsilon})} T' \leq \epsilon T'.$$

Do đó $f'(S_u^l) \geq g(S_u^l) \geq (1-\epsilon)T'$. Điều này cũng suy ra tồn tại một vòng lặp i, $1 \leq i \leq l$ sao cho $g(S_u^i) \geq (1-\epsilon)T'$ và $g(S_u^{i-1}) < (1-\epsilon)T'$.

Ký hiệu $S_u^{<e}$ là S_u trước khi thêm e vào S_u. Theo quy tắc lựa chọn phần tử mới vào S_u^i, ta có:

$$
\begin{aligned}
g(S_u^i) - g(S_u^{i-1}) &= \sum_{e \in S_u^i \setminus S_u^{i-1}} g(S_u^{<e}|e) \\
&\geq \sum_{e \in S_u^i \setminus S_u^{i-1}} c(e) \cdot \frac{(1-\epsilon)}{u}(T' - g(S_u^{i-1})) \\
&= c(S_u^i \setminus S_u^{i-1})\frac{(1-\epsilon)}{u}(T' - g(S_u^{i-1})). \qquad (4.1)
\end{aligned}
$$

Biến đổi lại bất đẳng thức (4.1) thu được

$$
\begin{aligned}
T' - g(S_u^i) &\leq \left(1 - \frac{(1-\epsilon)c(S_u^i \setminus S_u^{i-1})}{u}\right)\left(T - g(S_u^{i-1})\right) \\
&\leq e^{-\frac{(1-\epsilon)c(S_u^i \setminus S_u^{i-1})}{u}}(T - g(S_u^{i-1})).
\end{aligned}
$$

Bằng cách áp dụng liên tiếp bất đẳng thức trên, ta có

$$
\begin{aligned}
\epsilon T' < T' - g(S_u^{i-1}) &\leq e^{-\frac{(1-\epsilon)c(S_u^i \setminus S_u^{i-1})}{u}}(T' - f(S_u^{i-1})) \\
&\leq e^{-\frac{(1-\epsilon)c(S_u^i \setminus S_u^{i-1})}{u}} e^{-\frac{(1-\epsilon)c(S_u^{i-1} \setminus S_u^{i-2})}{u}}(T - f(S_u^{i-2})) \\
&= e^{-\frac{(1-\epsilon)c(S_u^i \setminus S_u^{i-2})}{u}}(T - f(S_u^{i-2})) \\
&\ldots \leq e^{-\frac{(1-\epsilon)c(S_u^{i-1})}{u}} T \leq e^{-\frac{(1-\epsilon)c(S_u^{i-1})}{(1+\epsilon)\mathsf{opt}'}} T'.
\end{aligned}
$$

Do đó $c(S_u^{i-1}) \leq \frac{1+\epsilon}{1-\epsilon}\ln(\frac{1}{\epsilon})\mathsf{opt}'$. Từ (4.1) và định nghĩa của $f(\cdot)$, ta có cận trên của $c(S_u^i \setminus S_u^{i-1})$, cụ thể là:

$$c(S_u^i \setminus S_u^{l-1}) \leq \frac{g(S_u^i) - g(S_u^{l-1})}{T' - g(S_u^{i-1})}\frac{u}{1-\epsilon} \leq \frac{1+\epsilon}{1-\epsilon}\mathsf{opt}'.$$

Do điều kiện tại dòng 21, từ vòng lặp i trở đi, không có phần tử mới nào được thêm vào S_u, điều này dẫn đến $S_u^l = S_u^i$. Do đó $f(S_u^l) \geq g(S_u^l) = g(S_u^i) \geq (1-\epsilon)T'$ và

$$c(S_u^l) = c(S_u^i) = c(S_u^{i-1}) + c(S_u^i \setminus S_u^{i-1}) \leq \Big(1 + \ln(\frac{1}{\epsilon})\Big)\frac{1+\epsilon}{1-\epsilon}\mathsf{opt}'.$$

Điều này hoàn tất chứng minh. □

Định lý 4.3. *Thuật toán 15 cần $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$ lượt quét, sử dụng $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$ bộ nhớ, có độ phức tạp truy vấn là $O\left(\log\left(\frac{1}{\epsilon}\right)\frac{n}{\epsilon^2}\log\left(\frac{n}{\epsilon}\right)\right)$ và trả về một nghiệm S sao cho: $f(S) \geq (1-\epsilon)T$ và $c(S) \leq \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon}\mathsf{opt}$.*

Chứng minh. **Chứng minh độ phức tạp.** Lượt quét đầu tiên cần $O(n)$ truy vấn và k bộ nhớ để tìm L. Lượt quét thứ hai sử dụng $|V_0|$ bộ nhớ và $O(n)$ truy vấn để tìm V_0 và C. Vì V_0 không phải là một nghiệm khả thi nên $|V_0| < k$. Thuật toán sau đó quét $\frac{1}{\epsilon}\log(\frac{1}{\epsilon})$ lần trên V (từ Dòng 17–23 trong Thuật toán 15). Với mỗi phần tử đầu vào, thuật toán thực hiện nhiều nhất

$$\begin{aligned}
|U| &\leq \log_{1+\epsilon}(\frac{C}{c'_0}) + 1 \leq \log_{1+\epsilon}(n\frac{c'_{max}}{c'_{min}}) + 1 \\
&\leq \log_{1+\epsilon}(n\frac{n\cdot j}{\epsilon}) + 1 \leq \log_{1+\epsilon}(\frac{n^3}{\epsilon}) + 1 \\
&= \frac{\log(\frac{n^3}{\epsilon})}{\log(1+\epsilon)} + 1 \leq \frac{1}{\epsilon}\log(\frac{n^3}{\epsilon}) + 1 = O\Big(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})\Big)
\end{aligned}$$

truy vấn. Do đó, tổng số truy vấn mà thuật toán thực hiện nhiều nhất là:

$$O(n) + O(n) + \frac{1}{\epsilon}\log(\frac{1}{\epsilon})\cdot n \cdot O\Big(\frac{1}{\epsilon}\log(\frac{n}{\epsilon})\Big) = O\Big(\log(\frac{1}{\epsilon})\frac{n}{\epsilon^2}\log(\frac{n}{\epsilon})\Big).$$

Với mỗi phần tử $u \in U$, theo lập luận tương tự như trong chứng minh Định lý 4.2, thuật toán cần nhiều nhất k bộ nhớ để lưu trữ S_u. Do đó, độ phức tạp bộ nhớ của thuật toán là:

$$O(k) + O(k) + k\cdot|U| = O(k) + O\Big(\frac{k}{\epsilon}\log(\frac{n}{\epsilon})\Big) = O\Big(\frac{k}{\epsilon}\log(\frac{n}{\epsilon})\Big).$$

Chứng minh bảo đảm xấp xỉ. Thuật toán 15 và Thuật toán 14 có cùng lượt quét thứ nhất và thứ hai. Theo lựa chọn V_0 và Bổ đề 4.1, ta có:

$$c(V_0) \leq |V_0|c'_{min} \leq n\frac{\epsilon c(u_j)}{n} \leq \epsilon c(v_j) \leq \epsilon\mathsf{opt}.$$

Theo Bổ đề 4.2, tồn tại một số nguyên $u \in U$ sao cho $f'(S_u) \geq T'(1-\epsilon)$ và $c(S_u) \leq \Big(1 + \ln(\frac{1}{\epsilon})\Big)\frac{1+\epsilon}{1-\epsilon}\mathsf{opt}'$. Theo quy tắc lựa chọn S tại Dòng 24 trong Thuật toán 15, luôn tồn tại S sao cho $f'(S) \geq (1-\epsilon)T'$ và $c(S) \leq c(S_u)$. Do đó:

$$\begin{aligned}
f(S) &= f(S_u \cup V_0) - f(V_0) + f(V_0) \\
&= f'(S_u) + f(V_0) \geq (1-\epsilon)T' + f(V_0) \\
&= (1-\epsilon)(T - f(V_0)) + f(V_0)
\end{aligned}$$

$$= (1-\epsilon)T + \epsilon f(V_0) \geq (1-\epsilon)T.$$

Vì $\mathsf{opt}' \leq \mathsf{opt}$, ta thu được

$$c(S) = C(V_0 \cup S_u) \leq \epsilon\mathsf{opt} + \left(1 + \ln(\frac{1}{\epsilon})\right)\frac{1+\epsilon}{1-\epsilon}\mathsf{opt}' < \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon}\mathsf{opt},$$

từ đó hoàn tất chứng minh. □

4.3. Thực nghiệm

4.3.1. Ứng dụng và bộ dữ liệu

Ngưỡng doanh thu (Revenue Threshold - RT) Dựa trên khái niệm bài toán Tối đa doanh thu *(Định nghĩa 1.26)*, bài toán RT được phát biểu như sau: Cho một mạng xã hội được biểu diễn bởi đồ thị $G = (V, E)$, trong đó tập đỉnh V đại diện cho người dùng trong mạng và tập cạnh E mô tả các mối quan hệ giữa người dùng. Hàm doanh thu quảng cáo được định nghĩa như sau:

$$f(X) = \sum_{u \in V}\left(\sum_{v \in X} w_{uv}\right)^{\alpha_u}$$

với:

- $X \subseteq V$ là tập con người dùng được chọn.
- w_{uv} là trọng số của cạnh giữa đỉnh u và v.
- α_u là hệ số trong khoảng $(0, 1)$, được gán ngẫu nhiên cho mỗi người dùng u.

Theo nghiên cứu của Kuhnle [74], hàm $f(\cdot)$ là một hàm đơn điệu và submodular. Không giống như mục tiêu trong bài toán Tối đa doanh thu, Bài toán RT nhằm tìm tập người dùng $X \subseteq V$ sao cho doanh thu vượt ngưỡng T, đồng thời tối thiểu chi phí quảng cáo.

Ứng dụng này sử dụng ba bộ dữ liệu từ SNAP[2], bao gồm: Facebook, GrQc và Enron. Bộ dữ liệu Facebook ($4,039$ đỉnh và $88,234$ cạnh) và GrQc ($5,242$ đỉnh và $14,496$ cạnh) có quy mô nhỏ, trong khi Enron có quy mô lớn hơn ($36,692$ đỉnh và $183,831$ cạnh), được thể hiện trong Bảng 4.3.

Ngưỡng phủ (Coverage Threshold - CT) Theo bài toán Tối đa độ phủ (Maximum Coverage) [74], ứng dụng CT được mô tả như sau: Cho đồ thị $G = (V, E)$, với tập đỉnh V và tập cạnh E. Với mỗi tập con $X \subseteq V$, định nghĩa tập X^I là tập các đỉnh có liên kết với ít nhất một đỉnh trong X. Hàm phủ được định nghĩa như sau:

$$f(X) = |X^I|.$$

Theo Kuhnle [74], hàm $f(\cdot)$ là một hàm đơn điệu và submodular. Mục tiêu của CT là tìm tập đỉnh $X \subseteq V$ sao cho kích thước của X^I vượt qua ngưỡng cho trước T, đồng thời tối thiểu chi phí liên quan.

[2]https://snap.stanford.edu/data/

Luận án sử dụng thêm ba bộ dữ liệu từ SNAP: Stanford, Hept và AstroPh. Trong đó, bộ dữ liệu Stanford có kích thước lớn nhất với $281,903$ đỉnh và $2,312,497$ cạnh, được biểu diễn dưới dạng đồ thị có hướng. Hai bộ dữ liệu Hept ($34,546$ đỉnh, $421,578$ cạnh) và AstroPh ($18,772$ đỉnh, $198,110$ cạnh) là đồ thị vô hướng. Thông tin chi tiết được trình bày trong Bảng 4.3.

Bảng 4.3: Bảng thống kê các bộ dữ liệu sử dụng trong thực nghiệm cho bài toán MSC

Ứng dụng	**Bộ dữ liệu**	**Số đỉnh**	**Số cạnh**	**Loại đồ thị**
Ngưỡng doanh thu	Facebook	$4,039$	$88,234$	Vô hướng
Ngưỡng doanh thu	GrQc	$5,242$	$14,496$	Vô hướng
Ngưỡng doanh thu	Enron	$36,692$	$183,831$	Vô hướng
Ngưỡng phủ	AstroPh	$18,772$	$198,110$	Vô hướng
Ngưỡng phủ	Hept	$34,546$	$421,578$	Vô hướng
Ngưỡng phủ	Stanford	$281,903$	$2,312,497$	Có hướng

Các bộ dữ liệu trong Bảng 4.3 được lựa chọn nhằm bao phủ nhiều quy mô và cấu trúc mạng khác nhau: từ mạng nhỏ như Facebook và GrQc, mạng trung bình như AstroPh, Hept và Enron, đến mạng lớn Stanford; đồng thời bao gồm cả đồ thị vô hướng và đồ thị có hướng. Hai nhóm ứng dụng RT và CT cũng phản ánh hai cách sử dụng hàm submodular đơn điệu trong thực nghiệm: một nhóm dựa trên hàm doanh thu và một nhóm dựa trên hàm phủ. Tuy vậy, các bộ dữ liệu này vẫn là các benchmark học thuật phổ biến từ SNAP, nên kết quả thực nghiệm chủ yếu cho thấy hành vi của thuật toán trong môi trường mô phỏng có kiểm soát, chưa thay thế cho đánh giá trên dữ liệu triển khai thực tế của từng lĩnh vực ứng dụng.

4.3.2. Thuật toán so sánh

Luận án so sánh các thuật toán đề xuất với các thuật toán hiện có phù hợp và có thể áp dụng cho bài toán MSC, được liệt kê như sau:

- **MULTI** [26]: Thuật toán giải bài toán MSC với đảm bảo xấp xỉ hai tiêu chí là $\big((1+\epsilon)(1+\frac{4}{\epsilon^2}), 1-\epsilon\big)$, thực hiện $O(\frac{\log(\mathsf{opt})}{c_{min}})$ lượt quét qua dữ liệu, thực hiện $O\big(\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\epsilon\log(1+\epsilon)}(n+\frac{\mathsf{opt}}{\epsilon c_{min}})\big)$ truy vấn và lưu trữ các nghiệm ứng viên với tổng chi phí không vượt quá $(1+\epsilon)(\frac{4}{\epsilon^2}+1)\mathsf{opt}$.

- **SINGLE** [26]: Thuật toán đạt cùng đảm bảo xấp xỉ hai tiêu chí như MULTI, yêu cầu $O\Big(\frac{nB\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\epsilon^2 c_{min}\log(1+\epsilon)}\Big)$ truy vấn, với tổng chi phí các nghiệm ứng viên bị chặn bởi $B(1+\epsilon)(\frac{4}{\epsilon^2}+1)\frac{\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\log(1+\epsilon)}$.

- **GREEDY** [49]: Thuật toán đạt tỉ lệ xấp xỉ hai tiêu chí là $(\log(\frac{1}{\epsilon}), 1-\epsilon)$.

Ba thuật toán GREEDY, SINGLE và MULTI được chọn làm thuật toán so sánh vì chúng đại diện cho các hướng tiếp cận gần nhất với bài toán MSC. GREEDY là chuẩn tham chiếu cổ điển, có chất lượng nghiệm tốt nhưng cần nhiều lượt quét dữ liệu. SINGLE và MULTI là các thuật toán luồng gần đây cho MSC, phù hợp để đánh giá trực tiếp lợi thế của các thuật toán đề xuất về số lượt quét, số truy vấn và bộ nhớ. Do đó, nhóm baseline này cho phép so sánh cả với phương pháp tham lam truyền thống và với các thuật toán luồng hiện có.

Nghiên cứu sinh thay đổi ngưỡng T trong tập giá trị $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ của $f(V)$, tương tự như nghiên cứu trước đó [26]. Tham số $\epsilon = 0.1$ được giữ cố định cho tất cả các thuật toán. Các thực nghiệm được thực hiện trên máy chủ HPC (partition = large partition, số luồng CPU = 32, số node = 2, bộ nhớ tối đa = $3,073$ GB). Các thuật toán được cài đặt bằng ngôn ngữ C++, sử dụng thư viện Chrono để đo thời gian thực thi và thư viện malloc để đo lường việc cấp phát bộ nhớ.

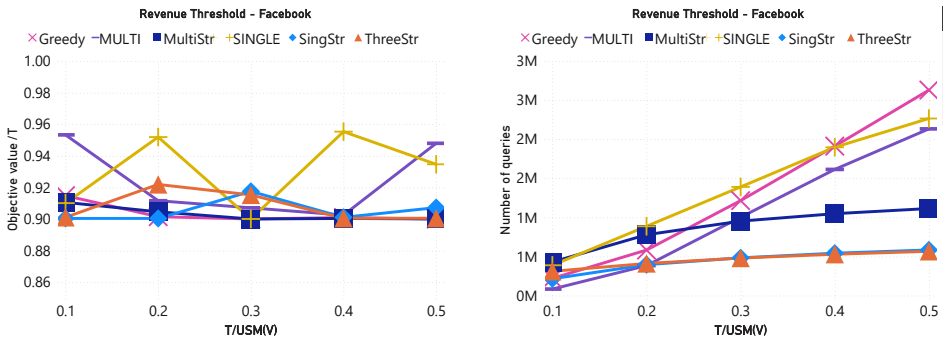
Bảng 4.4: Thiết lập thực nghiệm cho bài toán MSC

Ứng dụng	**Bộ dữ liệu**	**Thuật toán so sánh**	**Tham số**	**Chỉ tiêu đánh giá**
Ngưỡng doanh thu (RT)	Facebook, GrQc, Enron	SingStr, ThreeStr, MultiStr, GREEDY, SINGLE, MULTI	5 mức ngưỡng từ $0.1f(V)$ đến $0.5f(V)$; $\epsilon = 0.1$	Giá trị hàm mục tiêu, chi phí nghiệm, số truy vấn, bộ nhớ
Ngưỡng phủ (CT)	AstroPh, Hept, Stanford	SingStr, ThreeStr, MultiStr, GREEDY, SINGLE, MULTI	5 mức ngưỡng từ $0.1f(V)$ đến $0.5f(V)$; $\epsilon = 0.1$	Giá trị hàm mục tiêu, chi phí nghiệm, số truy vấn, bộ nhớ, số lượt quét

4.3.3. Kết quả thực nghiệm

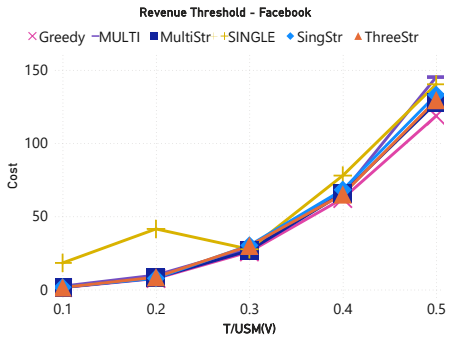
Kết quả thực nghiệm cho ứng dụng Ngưỡng Doanh thu (RT) được minh họa trong Hình 4.4, 4.5 và 4.6. Kết quả cho ứng dụng Ngưỡng Phủ (CT) được thể hiện trong Hình 4.7, 4.8 và 4.9.

Ngưỡng Doanh thu. Trước tiên, kết quả về giá trị mục tiêu được trình bày trong các Hình 4.4a, 4.5a và 4.6a, trong đó các giá trị $f(\cdot)$ được chuẩn hóa theo ngưỡng T. Tất cả các thuật toán đều cho giá trị hàm $f(\cdot)$ lớn hơn $(1-\epsilon)T$. Các thuật toán SingStr, ThreeStr, MultiStr và GREEDY cho thấy giá trị $f(\cdot)$ ổn định gần với $(1-\epsilon)T$, trong khi các thuật toán MULTI và SINGLE cho giá trị cao hơn nhưng không ổn định.

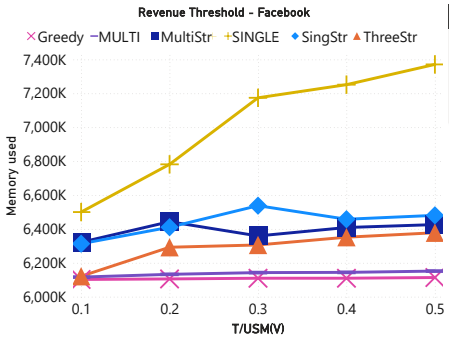


(a) Giá trị hàm mục tiêu

(b) Số lượng truy vấn



(c) Giá trị chi phí

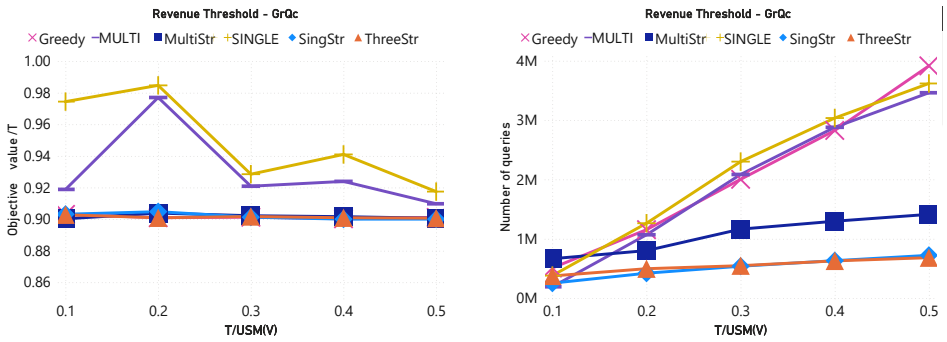


(d) Bộ nhớ sử dụng

Hình 4.4: Kết quả thực nghiệm trên tập dữ liệu Facebook cho ứng dụng ngưỡng doanh thu.

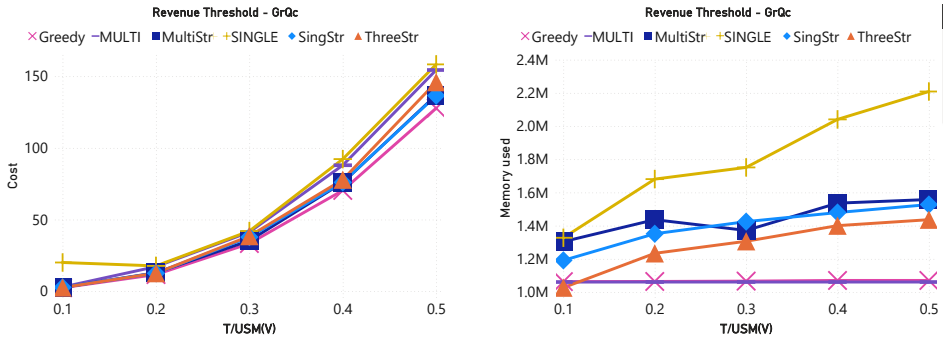
Tiếp theo, xét về chi phí (Hình 4.4c, 4.5c và 4.6c), các thuật toán cho kết quả tương tự nhau trên các tập dữ liệu. SingStr, ThreeStr, MultiStr và GREEDY luôn cho giá trị chi phí thấp, trong khi MULTI và SINGLE có chi phí cao hơn. Đặc biệt, trên tập Facebook, thuật toán SINGLE có chi phí cao gấp 5 đến 12 lần so với các thuật toán còn lại ở các mức ngưỡng $T = 0.1$ và 0.2. Trên các tập GrQc và Enron, MULTI và SINGLE duy trì chi phí cao hơn từ 1.5 đến 7.2 lần so với các thuật toán khác. Đáng chú ý, MultiStr và GREEDY luôn đạt chi phí thấp nhất trên tất cả các tập.

Về tổng thể, xét theo giá trị mục tiêu và chi phí, các thuật toán đề xuất đạt chất lượng nghiệm cạnh tranh với GREEDY và thường tiết kiệm chi phí hơn MULTI, SINGLE trong các thiết lập được khảo sát. Kết quả này phù hợp với mục tiêu thiết kế của các thuật toán luồng: duy trì chất lượng bao phủ gần ngưỡng yêu cầu trong khi kiểm soát số truy vấn và chi phí nghiệm.



(a) Giá trị hàm mục tiêu

(b) Số lượng truy vấn



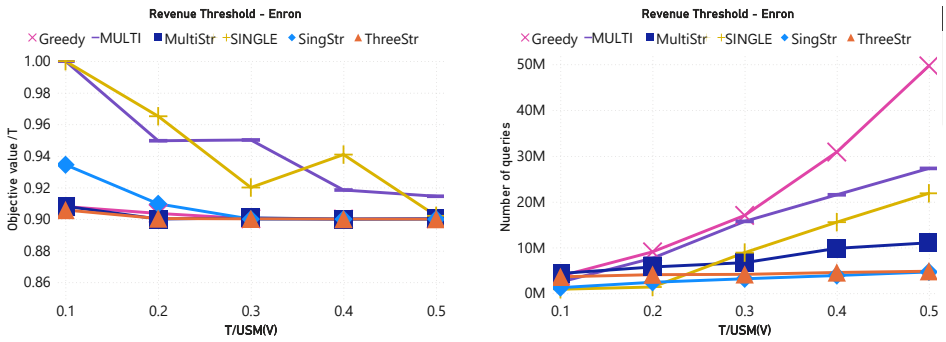
(c) Giá trị chi phí

(d) Bộ nhớ sử dụng

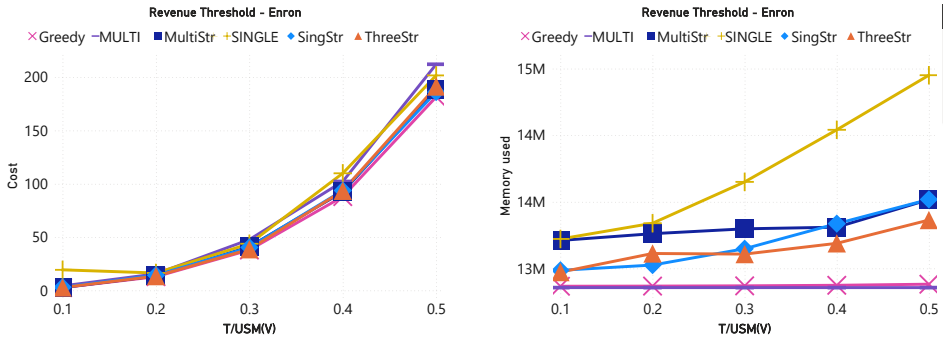
Hình 4.5: Kết quả thực nghiệm trên tập dữ liệu GrQc cho ứng dụng ngưỡng doanh thu.

Một tiêu chí quan trọng khác là số lượng truy vấn, vì nó gắn liền với thời gian thực thi. Kết quả truy vấn được trình bày trong Hình 4.4b, 4.5b và 4.6b. Với tập Facebook, SingStr và ThreeStr cho số lượng truy vấn thấp nhất, thấp hơn 1.9 đến 4.5 lần so với các thuật toán khác tại $T = 0.1$. MultiStr cần khoảng gấp đôi truy vấn của SingStr, trong khi GREEDY, MULTI và SINGLE cao hơn từ 3 đến 4.5 lần. Đặc biệt, khi ngưỡng T tăng, số lượng truy vấn của GREEDY, MULTI và SINGLE tăng nhanh, trong khi SingStr, ThreeStr và MultiStr giữ ổn định. Trên tập GrQc và Enron, xu hướng tương tự được duy trì; GREEDY cần nhiều truy vấn nhất, còn SingStr, ThreeStr, MultiStr giữ ở mức thấp và ổn định.

Cuối cùng, về sử dụng bộ nhớ (Hình 4.4d, 4.5d và 4.6d), các thuật toán được chia làm ba nhóm: sử dụng bộ nhớ thấp, trung bình và cao. GREEDY và MULTI tiêu tốn ít bộ nhớ nhất. SingStr, ThreeStr và MultiStr ở mức trung bình, với ThreeStr thấp nhất và MultiStr cao nhất trong nhóm này. SINGLE sử dụng nhiều bộ nhớ nhất trên mọi tập.

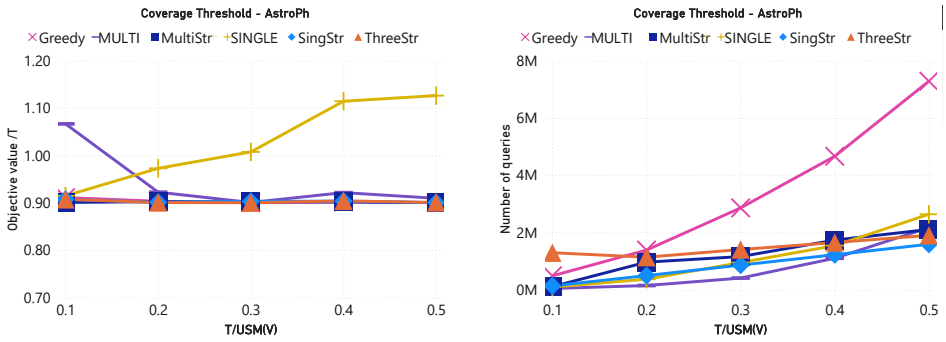


(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn

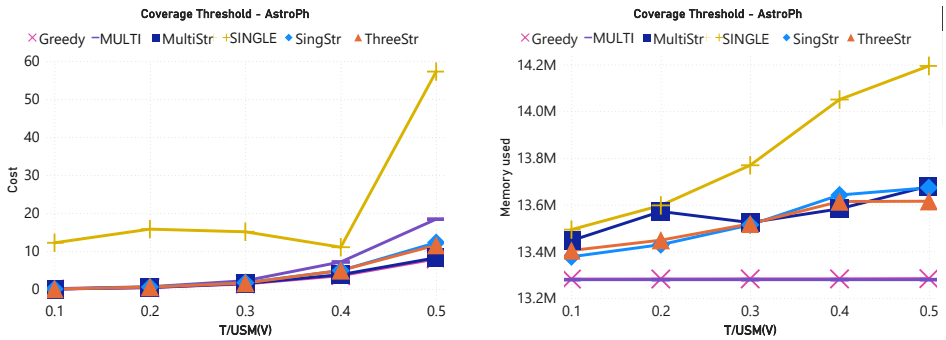


(c) Giá trị chi phí (d) Bộ nhớ sử dụng

Hình 4.6: Kết quả thực nghiệm trên tập dữ liệu Enron cho ứng dụng ngưỡng doanh thu.

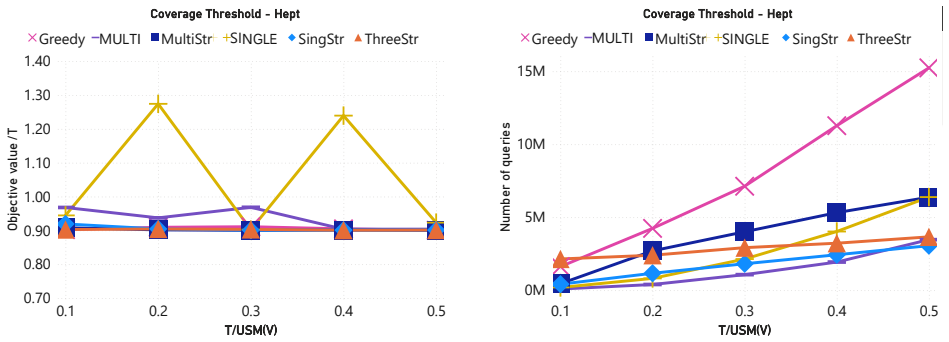


(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn

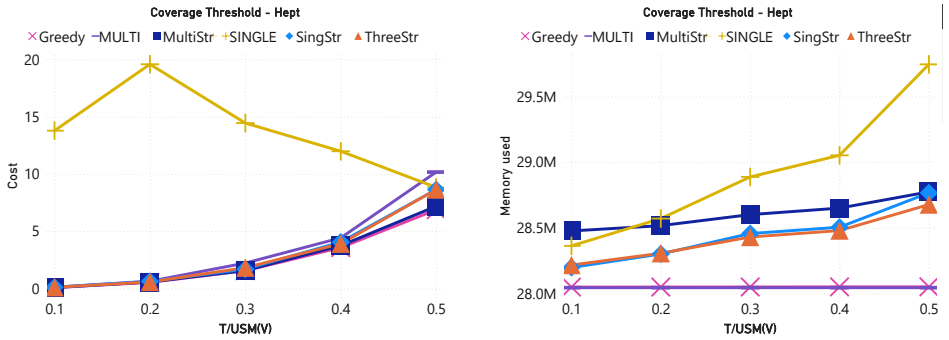


(c) Giá trị chi phí (d) Bộ nhớ sử dụng

Hình 4.7: Kết quả thực nghiệm trên tập dữ liệu AstroPh cho ứng dụng ngưỡng phủ.

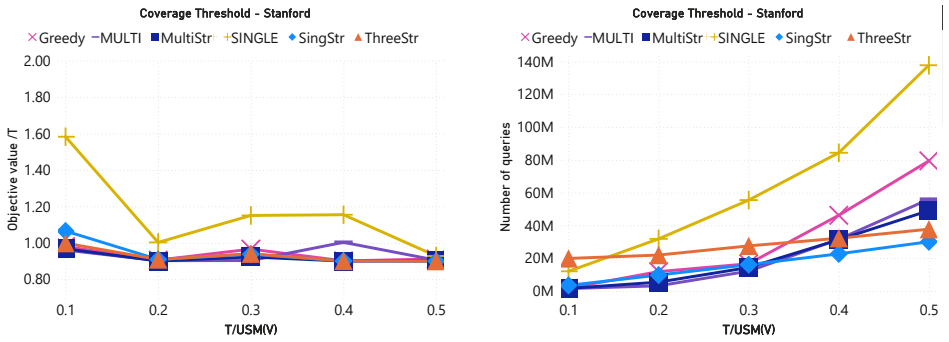


(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn

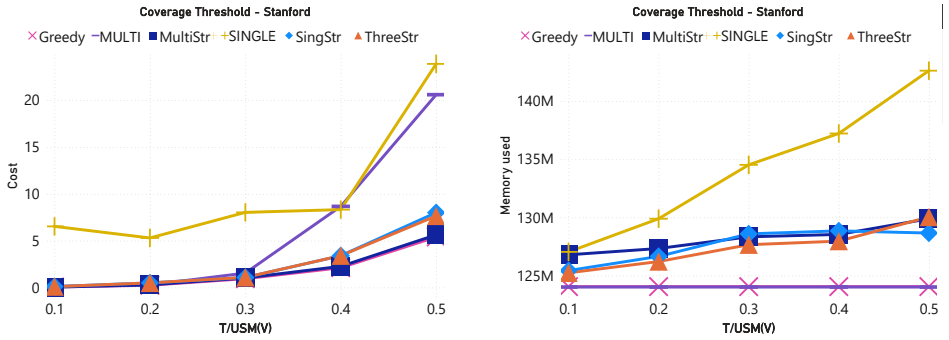


(c) Giá trị chi phí (d) Bộ nhớ sử dụng

Hình 4.8: Kết quả thực nghiệm trên tập dữ liệu Hept cho ứng dụng ngưỡng phủ.



(a) Giá trị hàm mục tiêu (b) Số lượng truy vấn



(c) Giá trị chi phí (d) Bộ nhớ sử dụng

Hình 4.9: Kết quả thực nghiệm trên tập dữ liệu Stanford cho ứng dụng ngưỡng phủ.

Ngưỡng Phủ. Tương tự như RT, các giá trị mục tiêu được thể hiện trong Hình 4.7a, 4.8a và 4.9a. Tất cả thuật toán đạt $f(\cdot) \geq (1-\epsilon)T$. Các thuật toán SingStr, ThreeStr, MultiStr và GREEDY cho kết quả ổn định gần $(1-\epsilon)T$, trong khi MULTI và SINGLE biến động mạnh hơn.

Về chi phí (Hình 4.7c, 4.8c, 4.9c), các thuật toán SingStr, ThreeStr, MultiStr, MULTI và GREEDY có chi phí tương đương nhau và thấp. SINGLE có chi phí cao hơn do thường chọn nghiệm có giá trị $f(\cdot)$ cao hơn mức ngưỡng cần thiết. Cụ thể, SINGLE có chi phí cao nhất ở cả ba tập AstroPh, Hept và Stanford, trong khi MultiStr và GREEDY thường xuyên đạt chi phí thấp nhất.

Xét số lượng truy vấn (Hình 4.7b, 4.8b và 4.9b), GREEDY cần nhiều truy vấn nhất trên tập AstroPh. MULTI và SINGLE chỉ có truy vấn thấp hơn SingStr, ThreeStr và MultiStr với T nhỏ, nhưng tăng nhanh khi T lớn. Kết quả này cho thấy SingStr, ThreeStr và MultiStr có xu hướng ổn định hơn khi ngưỡng tăng dần trong các bộ dữ liệu được khảo sát.

Trên tập Hept, xu hướng vẫn giữ nguyên: GREEDY cao nhất, SingStr và MULTI thấp nhất. MULTI và SINGLE tăng nhanh hơn so với SingStr, ThreeStr và MultiStr khi T tăng. Với tập Stanford, SINGLE có số truy vấn thấp nhất, nhưng chỉ ở các mức T nhỏ; khi T tăng, các thuật toán SingStr, ThreeStr và MultiStr thể hiện ưu thế về sự ổn định.

Cuối cùng, về bộ nhớ (Hình 4.7d, 4.8d, 4.9d), GREEDY và MULTI tiêu tốn bộ nhớ thấp, SingStr, ThreeStr và MultiStr ở mức trung bình. SINGLE tiếp tục là thuật toán ngốn bộ nhớ nhiều nhất. Khi T tăng, bộ nhớ của SINGLE tăng nhanh, trong khi các thuật toán còn lại giữ mức ổn định.

Bảng 4.5: Bảng so sánh số lượt quét của các thuật toán trên tập Stanford

T	GREEDY	SINGLE	MULTI	SingStr	ThreeStr	MultiStr
0.1	6	1	58	1	3	5
0.2	42	1	66	1	3	6
0.3	59	1	80	1	3	7
0.4	164	1	90	1	3	7
0.5	282	1	96	1	3	8

Ngoài ra, Bảng 4.5 trình bày số lượt quét mà các thuật toán thực hiện trên tập dữ liệu Stanford. Cả về lý thuyết và thực nghiệm, SINGLE, SingStr và ThreeStr giữ số lượt quét ổn định (1 và 3 lượt tương ứng). GREEDY tăng dần theo kích thước nghiệm và tăng nhanh khi T lớn. MULTI luôn cần số lượt quét cao (từ 58 đến 96), cao hơn nhiều so với MultiStr (chỉ 5 đến 8 lượt). Số lượt quét của MultiStr tăng chậm hơn nhiều so với tốc độ tăng của T trong thiết lập này.

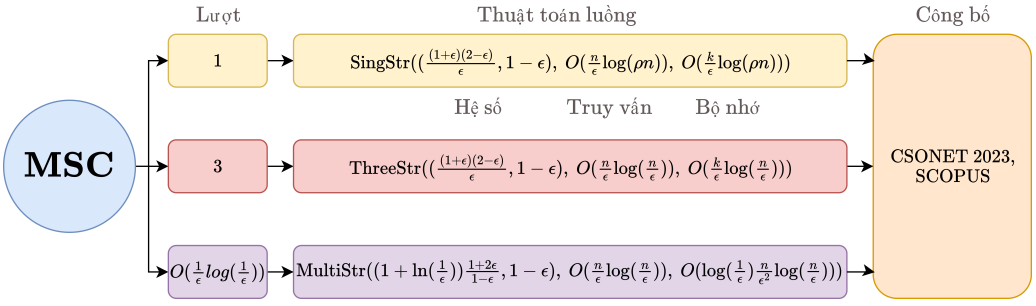
Bảng 4.6: Tổng hợp nhận xét chính từ thực nghiệm cho bài toán MSC

Nhóm thuật toán	**Chất lượng phủ**	**Chi phí nghiệm**	**Số truy vấn/lượt quét**	**Bộ nhớ và phạm vi phù hợp**
SingStr	Đạt ngưỡng yêu cầu và ổn định gần $(1-\epsilon)T$.	Cạnh tranh với GREEDY và thường thấp hơn SINGLE, MULTI trong RT.	Một lượt quét; số truy vấn ổn định khi T tăng.	Phù hợp khi dữ liệu chỉ có thể quét một lần; bộ nhớ ở mức trung bình.
ThreeStr	Tương tự SingStr trong các thiết lập khảo sát.	Cạnh tranh với GREEDY và MultiStr.	Ba lượt quét; truy vấn thấp và ổn định trên nhiều bộ dữ liệu.	Phù hợp khi có thể dùng thêm lượt quét để giảm chi phí truy vấn.
MultiStr	Đạt ngưỡng yêu cầu, hướng tới chất lượng gần GREEDY.	Thường thuộc nhóm chi phí thấp, đặc biệt trong CT.	Nhiều lượt quét nhưng thấp hơn MULTI trên Stanford.	Phù hợp khi chấp nhận thêm lượt quét để cải thiện hệ số xấp xỉ.
GREEDY, SINGLE, MULTI	Đạt ngưỡng, nhưng SINGLE và MULTI có thể vượt ngưỡng nhiều hơn mức cần thiết.	GREEDY thường thấp; SINGLE và MULTI có thể cao hơn trong một số thiết lập.	GREEDY/MULTI cần nhiều lượt quét hoặc truy vấn khi T tăng.	Là baseline quan trọng, nhưng có hạn chế khi xử lý dữ liệu luồng quy mô lớn.

4.4. Kết luận chương

Chương này đã tập trung nghiên cứu bài toán MSC – Bài toán nghiên cứu 4, một bài toán nền tảng trong tối ưu submodular với nhiều ứng dụng thực tiễn trong học máy và trí tuệ nhân tạo. Đóng góp nổi bật của chương là việc thiết kế và phân tích ba thuật toán luồng hiệu quả: SingStr, ThreeStr và MultiStr, được xây dựng hướng đến các kịch bản xử lý dữ liệu lớn theo luồng với tài nguyên tính toán và lưu trữ giới hạn.

Về mặt lý thuyết, các thuật toán được đề xuất đạt hệ số xấp xỉ hai tiêu chí cạnh tranh với thuật toán tham lam, đồng thời kiểm soát số lượt quét, độ phức tạp truy vấn và bộ nhớ sử dụng theo các mức đánh đổi khác nhau. Cụ thể, thuật toán SingStr chỉ cần một lượt quét, ThreeStr đạt độ phức tạp đa thức với ba lượt quét, còn MultiStr hướng tới hệ số xấp xỉ gần với thuật toán tham lam hơn khi cho phép nhiều lượt quét hơn (xem Hình 4.10).



Hình 4.10: Tổng hợp các kết quả chính của luận án cho bài toán MSC.

Về mặt thực nghiệm, các thuật toán được kiểm chứng trên hai ứng dụng: ngưỡng doanh thu và ngưỡng phủ, sử dụng sáu tập dữ liệu benchmark từ SNAP. Kết quả trong Bảng 4.6 cho thấy các thuật toán đề xuất đạt chất lượng nghiệm gần ngưỡng mong muốn, có số truy vấn ổn định khi ngưỡng tăng và thường có chi phí nghiệm cạnh tranh với các thuật toán so sánh. Tuy nhiên, phạm vi đánh giá vẫn dựa trên các bộ dữ liệu benchmark và hai mô hình ứng dụng mô phỏng, do đó kết quả cần được hiểu như bằng chứng thực nghiệm về tính hiệu quả trong môi trường kiểm soát, chưa thay thế cho đánh giá triển khai trên dữ liệu nghiệp vụ thực tế.

Những thuật toán này góp phần trả lời câu hỏi nghiên cứu được đặt ra ở đầu chương: có thể xây dựng các thuật toán luồng cho MSC với các bảo đảm xấp xỉ hai tiêu chí và chi phí tính toán phù hợp hơn cho dữ liệu lớn.

Các kết quả đạt được trong chương này bổ sung một hướng tiếp cận cho các bài toán tối ưu submodular trong môi trường dữ liệu lớn. Các kết quả nghiên cứu trong chương đã được công bố tại hội nghị quốc tế **CSONET 2023 (SCOPUS)**, thể hiện đóng góp học thuật của luận án:

1. **Tan D. Tran**, Canh V. Pham, Dung T. Pham, Uyen T. Nguyen, *Improved Streaming Algorithm for Minimum Cost submodular Cover Problem*, 2023, In Proceedings of the 12th International Conference on Computational Data and Social Networks (CSONET 2023), 222-233 **(SCOPUS)**;

KẾT LUẬN

Luận án tập trung nghiên cứu chủ đề các thuật toán xấp xỉ cho bài toán tối ưu hàm submodular và các hàm mở rộng dưới các ràng buộc khác nhau. Hướng tiếp cận cốt lõi là khai thác cấu trúc toán học đặc biệt của hàm submodular và các lớp liên quan như k-submodular và DR-submodular để thiết kế các thuật toán có đảm bảo lý thuyết rõ ràng về hệ số xấp xỉ, độ phức tạp truy vấn và độ phức tạp song song. Đồng thời, luận án chú trọng đến khả năng triển khai thực nghiệm trên dữ liệu thực, nhằm đánh giá hiệu quả thuật toán trong các bài toán có quy mô lớn, ràng buộc khắt khe và không gian nghiệm phức tạp. Quan điểm xuyên suốt là xây dựng các phương pháp cân bằng giữa hiệu năng lý thuyết và khả năng ứng dụng trong thực tiễn. Dựa trên khoảng trống lý thuyết được phân tích, luận án đã lựa chọn và nghiên cứu bốn bài toán tiêu biểu (xem Hình 5.1):

1. **Tối đa hàm submodular với ràng buộc chi phí**: luận án đã phát triển ba thuật toán DLA, RLA và AST với hệ số xấp xỉ lần lượt là $\frac{1}{6} - \epsilon$, $\frac{1}{4} - \epsilon$ và $\frac{1}{7} - \epsilon$. Các thuật toán này đạt các bảo đảm lý thuyết rõ ràng về độ phức tạp truy vấn và độ phức tạp song song, đồng thời cho kết quả thực nghiệm tích cực trong các môi trường tính toán tuần tự và song song được khảo sát. Các kết quả nghiên cứu đã được công bố tại hội nghị quốc tế International Joint Conference on Artificial Intelligence (IJCAI 2023, IJCAI 2024) – hai hội nghị được xếp hạng **A*** theo bảng xếp hạng hội nghị CORE[1].

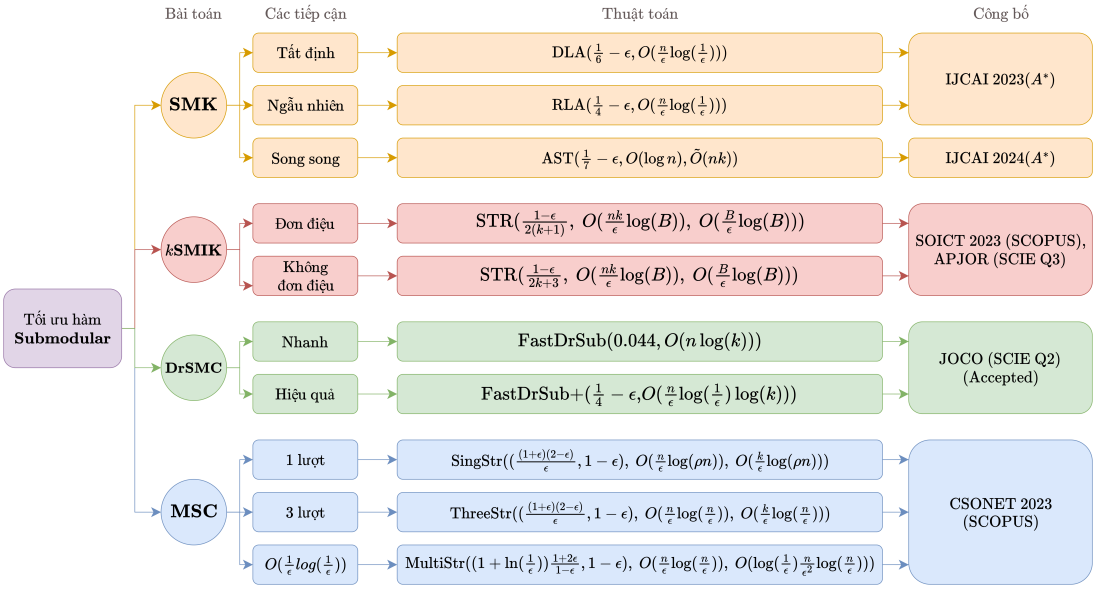
2. **Tối đa hàm k-submodular với ràng buộc chi phí nhóm**: luận án đề xuất thuật toán luồng STR, đạt hệ số xấp xỉ $\frac{1-\epsilon}{2(k+1)}$ trong trường hợp đơn điệu và $\frac{1-\epsilon}{2k+3}$ trong trường hợp không đơn điệu, với độ phức tạp truy vấn và bộ nhớ nhỏ. Thuật toán này phù hợp với môi trường dữ liệu lớn và bổ sung một hướng tiếp cận có bảo đảm lý thuyết cho bài toán tối ưu k-submodular trong bối cảnh ràng buộc chi phí nhóm. Kết quả nghiên cứu đã được trình bày tại The 12th International Symposium on Information and Communication Technology (SOICT 2023), kỷ yếu hội thảo được lập chỉ mục trong SCOPUS và được công bố trên Asia-Pacific Journal of Operational Research (APJOR), một tạp chí quốc tế thuộc danh mục SCIE, được xếp hạng **Q3**.

3. **Tối đa hàm DR-submodular với ràng buộc lực lượng**: luận án phát triển hai thuật toán mới là FastDrSub và FastDrSub+, lần lượt đạt hệ số xấp xỉ 0.044 và $\frac{1}{4} - \epsilon$ với độ phức tạp truy vấn thấp. Các thuật toán này cung cấp bảo đảm xấp xỉ hằng số cho bài toán được xét, đồng thời cho thấy hiệu quả rõ ràng trong thực nghiệm tối đa doanh thu với hàm mục tiêu DR-submodular. Kết quả nghiên cứu đã được công bố trên Journal of Combinatorial Optimization (JOCO), một tạp chí quốc tế thuộc danh mục SCIE, được xếp hạng **Q2**.

4. **Bài toán phủ submodular với chi phí tối thiểu**: luận án đã xây dựng ba thuật toán luồng hiệu quả là SingStr, ThreeStr và MultiStr, cho phép xử lý dữ liệu lớn theo

[1] https://portal.core.edu.au/conf-ranks/

luồng. Các thuật toán đạt hệ số xấp xỉ gần bằng tham lam nhưng sử dụng ít truy vấn và bộ nhớ hơn đáng kể, đồng thời thể hiện tính hiệu quả trong các ứng dụng như ngưỡng doanh thu và ngưỡng phủ. Các kết quả nghiên cứu đã được công bố tại International Conference on Computational Data and Social Networks (CSoNet 2023), kỷ yếu hội nghị được lập chỉ mục trong SCOPUS.



Hình 5.1: Tổng hợp các kết quả chính của luận án.

Bốn bài toán nghiên cứu đại diện cho sự đa dạng về loại hàm mục tiêu, tính đơn điệu và dạng ràng buộc, từ đó hình thành một bức tranh tổng thể về không gian bài toán tối ưu submodular hiện đại. Việc lựa chọn vừa bao phủ các trường hợp đã có kết quả nền để cải tiến, vừa khai phá các bài toán chưa từng được nghiên cứu, đảm bảo tính cân đối giữa chiều sâu và chiều rộng nghiên cứu. Về mặt lý thuyết, luận án đóng góp các kết quả mới có giá trị học thuật rõ ràng. Cụ thể, các thuật toán được đề xuất đều có bảo đảm xấp xỉ, trong đó một số trường hợp đạt hệ số tốt hoặc cải thiện các chỉ tiêu về truy vấn, bộ nhớ, số lượt quét hay độ phức tạp song song so với những phương pháp liên quan trong phạm vi bài toán tương ứng. Luận án cũng đưa ra các phân tích chi tiết về độ phức tạp song song và số lượng truy vấn cần thiết, hỗ trợ việc áp dụng cho các bài toán lớn. Các kết quả này đã được công bố tại nhiều hội nghị và tạp chí quốc tế uy tín có phản biện. Điều này thể hiện tính mới, độ tin cậy học thuật và tiềm năng mở rộng ứng dụng của các thuật toán tối ưu submodular được nghiên cứu trong luận án.

Bên cạnh các kết quả đạt được, luận án vẫn còn một số hạn chế cần tiếp tục được nghiên cứu. Thứ nhất, mặc dù các thuật toán đề xuất đều có bảo đảm lý thuyết rõ ràng, hệ số xấp xỉ trong một số bài toán, đặc biệt là đối với các lớp hàm mở rộng như k-submodular và DR-submodular dưới các ràng buộc phức tạp, vẫn còn khoảng cách để cải thiện. Thứ hai, việc giảm số lượng truy vấn, bộ nhớ hoặc số lượt duyệt dữ liệu trong các thuật toán luồng và thuật toán song song có thể dẫn đến sự đánh đổi nhất định

với chất lượng nghiệm hoặc phạm vi áp dụng của thuật toán. Thứ ba, các đánh giá thực nghiệm trong luận án chủ yếu tập trung vào một số mô hình ứng dụng và bộ dữ liệu tiêu biểu, do đó chưa phản ánh đầy đủ hiệu quả của các phương pháp trong các môi trường dữ liệu lớn, phân tán hoặc có ràng buộc thực tiễn đa dạng hơn. Những hạn chế này cũng chính là cơ sở để xác định các hướng phát triển tiếp theo của luận án.

Từ các kết quả và hạn chế nêu trên, luận án mở ra nhiều hướng nghiên cứu tiếp theo có tiềm năng phát triển cả về lý thuyết và ứng dụng. Trước hết, có thể tiếp tục cải thiện hệ số xấp xỉ, độ phức tạp truy vấn và độ phức tạp bộ nhớ của các thuật toán đã đề xuất, đặc biệt đối với các bài toán liên quan đến hàm k-submodular và DR-submodular. Bên cạnh đó, một hướng nghiên cứu quan trọng là mở rộng các phương pháp của luận án sang những lớp ràng buộc tổng quát hơn hoặc khác biệt hơn so với các ràng buộc đã xét, chẳng hạn như ràng buộc matroid, giao của nhiều matroid, knapsack nhiều chiều, ràng buộc công bằng, ràng buộc bao phủ, ràng buộc về độ đa dạng, ràng buộc ngân sách động hoặc các mô hình kết hợp nhiều loại ràng buộc. Ngoài ra, các thuật toán có thể được phát triển tiếp cho các mô hình dữ liệu hiện đại như dữ liệu luồng, dữ liệu phân tán, dữ liệu thay đổi theo thời gian và môi trường trực tuyến. Một hướng khác có nhiều tiềm năng là kết hợp tối ưu submodular với học máy, chẳng hạn như học hàm mục tiêu từ dữ liệu, chọn dữ liệu huấn luyện, tóm tắt dữ liệu lớn, hoặc thiết kế các phương pháp lai giữa học máy và tối ưu tổ hợp nhưng vẫn duy trì các bảo đảm lý thuyết. Những hướng nghiên cứu này không chỉ mở rộng phạm vi của các kết quả trong luận án mà còn góp phần thúc đẩy ứng dụng của tối ưu submodular trong các hệ thống dữ liệu lớn và các bài toán ra quyết định phức tạp.

DANH MỤC CÔNG TRÌNH KHOA HỌC LIÊN QUAN

1. Canh V. Pham, **Tan D. Tran**, Dung K.T. Ha, My T. Thai, *Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint*, 2023, In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023), 4127-4135 **(RANK A*)**;

2. **Tan D. Tran**, Canh V. Pham, Dung K.T. Ha, *Maximizing a k-submodular Maximization Function under an Individual Knapsack Constraint*, 2023, In Proceedings of the 12th International Symposium on Information and Communication Technology (SoICT 2023), 56-62 **(SCOPUS)**;

3. **Tan D. Tran**, Canh V. Pham, Dung T. Pham, Uyen T. Nguyen, *Improved Streaming Algorithm for Minimum Cost submodular Cover Problem*, 2023, In Proceedings of the 12th International Conference on Computational Data and Social Networks (CSONET 2023), 222-233 **(SCOPUS)**;

4. **Tan D. Tran**, Canh V. Pham, Dung K.T. Ha, Phuong N.H. Pham, *Improved parallel algorithm for non-monotone submodular maximization under knapsack constraint*, 2024, In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024), 1961-1969 **(RANK A*)**;

5. **Tan D. Tran**, Canh V. Pham, Dung K.T. Ha, *k-submodular Maximization Under Individual Knapsack Constraints: Applications and Streaming Algorithm*, 2025, Asia-Pacific Journal of Operational Research **(SCIE, Q3)**.

6. **Tan D. Tran**, Canh V. Pham, *Fast Approximation Algorithm for Non-Monotone DR-submodular Maximization under Size Constraint*, 2025, Journal of Combinatorial Optimization **(SCIE, Q2)**.

TÀI LIỆU THAM KHẢO

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29, 1996.

[2] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Rebecca Reiffenhäuser. Submodular maximization subject to a knapsack constraint: Combinatorial algorithms with near-optimal adaptive complexity. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 231–242, 2021.

[3] Georgios Amanatidis, Federico Fusco, Philip Lazos, Stefano Leonardi, and Rebecca Reiffenhäuser. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 33, pages 16903–16915, 2020.

[4] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.

[5] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 671–680, 2014.

[6] Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1497–1514. SIAM, 2014.

[7] Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1138–1151. ACM, 2018.

[8] Judit Bar-Ilan, Guy Kortsarz, and David Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250(1-2):179–200, 2001.

[9] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint*, arXiv:1611.09940, 2016.

[10] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[11] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[12] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R Thibaux. Intel lab. 2004.

[13] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms*, 14(3):32:1–32:20, 2018.

[14] Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Mathematical Operations Research*, 44(3):988–1005, 2019.

[15] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. SIAM, 2014.

[16] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms 2015*, pages 1149–1168, 2015.

[17] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[18] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[19] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

[20] Jingwen Chen, Zhongzheng Tang, and Chenhao Wang. Monotone k-submodular knapsack maximization: An analysis of the greedy+ singleton algorithm. In *International Conference on Algorithmic Applications in Management*, pages 144–155. Springer, 2022.

[21] Lin Chen, Moran Feldman, and Amin Karbasi. Unconstrained submodular maximization with constant adaptive complexity. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 102–113, 2019.

[22] Yixin Chen and Alan Kuhnle. Approximation algorithms for size-constrained non-monotone submodular maximization in deterministic linear time. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 250–261. ACM, 2023.

[23] Yixin Chen and Alan Kuhnle. Practical and parallelizable algorithms for non-monotone submodular maximization with size constraint. *Journal of Artificial Intelligence Research*, 79:599–637, 2024.

[24] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. In *Proceedings of the Symposium on New Directions and Recent Results in Algorithms and Complexity*, pages 441–446. Academic Press, 1976.

[25] Guojin Cong and David A Bader. The euler tour technique and parallel rooted spanning tree. In *International Conference on Parallel Processing, 2004. ICPP 2004.*, pages 448–457. IEEE, 2004.

[26] Victoria Crawford. Scalable bicriteria algorithms for non-monotone submodular cover. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 9517–9537. PMLR, 2023.

[27] Victoria Crawford, Alan Kuhnle, and My Thai. Submodular cost submodular cover with an approximate oracle. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1426–1435. PMLR, 2019.

[28] Victoria G. Crawford. Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1661–1667, 2021.

[29] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, and Zhiyu Li. Streaming algorithms for constrained submodular maximization. In *Proceedings of the ACM SIGMETRICS conference on Measurement and Analysis of Computer Systems*, volume 6, pages 54:1–54:32, 2021.

[30] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, and Aakas Zhiyuli. Practical parallel algorithms for submodular maximization subject to a knapsack constraint with nearly optimal adaptivity. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 7261–7269. AAAI Press, 2023.

[31] Shuang Cui, Kai Han, Jing Tang, He Huang, Xueying Li, Aakas Zhiyuli, and Hanxiao Li. Practical parallel algorithms for non-monotone submodular maximization. *CoRR*, abs/2308.10656, 2023.

[32] Ding-Zhu Du, Panos M Pardalos, Xiaodong Hu, and Weili Wu. *Introduction to combinatorial optimization*, volume 196. Springer Nature, 2022.

[33] Alina Ene and Huy Nguyen. Parallel algorithm for non-monotone dr-submodular maximization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2902–2911. PMLR, 2020.

[34] Alina Ene and Huy L. Nguyen. A reduction for optimizing lattice submodular functions with diminishing returns. *arXiv e-prints*, page 1606, 2016.

[35] Alina Ene and Huy L. Nguyen. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *LIPIcs*, pages 53:1–53:12, 2019.

[36] Alina Ene and Huy L. Nguyen. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 274–282. SIAM, 2019.

[37] Alina Ene and Huy L. Nguyen. Streaming algorithm for monotone k-submodular maximization with cardinality constraints. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 5944–5967. PMLR, 2022.

[38] Alina Ene, Huy L. Nguyen, and Adrian Vladu. Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 90–101. ACM, 2019.

[39] K. Erciyes. *Vertex Cover*. Springer London, 2013.

[40] Paul Erd6s and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[41] Matthew Fahrbach, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1833–1842. PMLR, 2019.

[42] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[43] Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[44] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 570–579, 2011.

[45] Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. In *Management Science*, volume 50, pages 1861–1871. 2004.

[46] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, November 1995.

[47] Ryan Gomes and Andreas Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning*, pages 391–398. Omnipress, 2010.

[48] Corinna Gottschalk and Britta Peis. Submodular function maximization on the bounded integer lattice. In *Proceedings of the 13th International Workshop on Approximation and Online Algorithms*, pages 133–144. Springer, 2015.

[49] Amit Goyal, Francesco Bonchi, Laks V. S. Lakshmanan, and Suresh Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3(2):179–192, 2013.

[50] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proceedings of the 6th International Workshop on Internet and Network Economics*, volume 6484, pages 246–257, 2010.

[51] Dung TK Ha, Canh V Pham, and Tan D Tran. Improved approximation algorithms for k-submodular maximization under a knapsack constraint. *Computers & Operations Research*, 161:106452, 2024.

[52] Kai Han, Zongmai Cao, Shuang Cui, and Benwei Wu. Deterministic approximation for submodular maximization over a matroid in nearly linear time. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[53] Kai Han, Shuang Cui, Tianshuai Zhu, Enpei Zhang, Benwei Wu, Zhizhuo Yin, Tong Xu, Shaojie Tang, and He Huang. Approximation algorithms for submodular data summarization with a knapsack constraint. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, volume 5, pages 1–31. ACM, 2021.

[54] Yijie Han. Parallel algorithms for linked list and beyond. In *Proceedings of the International Symposium on Algorithms*, pages 86–100. Springer, 1990.

[55] Mark Harris, Shubhabrata Sengupta, and John D Owens. Gpu gems 3, parallel prefix sum (scan) with cuda, chapter 39. *NVIDIA Corporation*, 2007.

[56] Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web*, page 189–198. Association for Computing Machinery, 2008.

[57] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[58] Chien-Chung Huang and Naonori Kakimura. Improved streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 83(3):879–902, 2021.

[59] Chien-Chung Huang and Naonori Kakimura. Multi-pass streaming algorithms for monotone submodular function maximization. *Theory of Computing Systems*, 66(1):354–394, 2022.

[60] Chien-Chung Huang, Naonori Kakimura, and Yuichi Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 82(4):1006–1032, 2020.

[61] Anna Huber and Vladimir Kolmogorov. Towards minimizing k-submodular functions. In *Proceedings of the 2nd International Symposium on Combinatorial Optimization*, pages 451–462. Springer, 2012.

[62] Hoàng Xuân Huấn and Đỗ Đức Đông. *Giáo trình Tối ưu hóa*. Đại học Quốc gia Hà Nội, Hà Nội, 2018.

[63] Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. Improved approximation algorithms for k-submodular function maximization. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 404–413. SIAM, 2016.

[64] Rishabh K Iyer and Jeff A Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. *Advances in Neural Information Processing Systems*, 26:2436–2444, 2013.

[65] Richard M Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2009.

[66] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[67] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[68] Natallia Kokash. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, 1:1–7, 2005.

[69] Bernhard Korte and Jens Vygen. *Combinatorial optimization: Theory and Algorithms*. Springer, 2008.

[70] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, volume 7, pages 1650–1654, 2007.

[71] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Technical Reports, University of Toronto*, 2019.

[72] Alan Kuhnle. Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pages 2371–2381, 2019.

[73] Alan Kuhnle. Nearly linear-time, parallelizable algorithms for non-monotone submodular maximization. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 8200–8208, 2021.

[74] Alan Kuhnle. Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 1360–1368. PMLR, 2021.

[75] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth*

annual ACM-SIAM symposium on Discrete algorithms, pages 545–554. SIAM, 2009.

[76] Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematical Operations Research*, 38(4):729–739, 2013.

[77] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. Fast greedy algorithms in mapreduce and streaming. In *Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 1–10. ACM, 2013.

[78] Lilly Kumari and Jeff Bilmes. Submodular span, with applications to conditional data summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12344–12352, 2021.

[79] Lei Lai, Qiufen Ni, Changhong Lu, Chuanhe Huang, and Weili Wu. Monotone submodular maximization over the bounded integer lattice with cardinality constraints. *Discrete Mathematics, Algorithms and Applications*, 11(06):1950075, 2019.

[80] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

[81] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2–es, March 2007.

[82] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, 2007.

[83] Jure Leskovec and Andrej Krevl. A. snap datasets: Stanford large network dataset collection. 2014.

[84] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[85] Wenxin Li. Nearly linear time algorithms and lower bound for submodular maximization. *preprint, arXiv:1804.08178*, 2018.

[86] Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. In *Proceedings of the 36th Conference on Neural Information Processing Systems*, pages 7887–7897, 2022.

[87] Yanfei Li, Min Li, Qian Liu, and Yang Zhou. Dr-submodular function maximization with adaptive stepsize. In *Proceedings of the 29th International Conference on Computing and Combinatorics*, pages 347–358. Springer, 2023.

[88] Yajing Liu, Edwin KP Chong, Ali Pezeshki, and Zhenliang Zhang. Submodular optimization problems and greedy strategies: A survey. *Discrete Event Dynamic Systems*, 30(3):381–412, 2020.

[89] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[90] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pages 1358–1367, 2016.

[91] Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, volume 28, 2015.

[92] Shanmugavelayutham Muthukrishnan et al. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005.

[93] George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular setfunction. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[94] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14:265–294, 1978.

[95] Phuong NH Pham, Quy TN Co, Anh N. Su, Phuong T. Pham, Canh V. Pham, and Vaclav Snasel. k-submodular cover problem: Application and algorithms. In *Proceedings of the 11th International Symposium on Information and Communication Technology*, pages 442–449, 2022.

[96] Rad Niazadeh, Tim Roughgarden, and Joshua R Wang. Optimal algorithms for continuous non-monotone submodular and dr-submodular maximization. *Journal of Machine Learning Research*, 21(125):1–31, 2020.

[97] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the International Conference on Machine Learning*, volume 80, pages 3826–3835, 2018.

[98] Naoto Ohsaka and Yuichi Yoshida. Monotone k-submodular function maximization with size constraints. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pages 694–702, 2015.

[99] Hiroki Oshima. Derandomization for k-submodular maximization. In *Proceedings of the 28th International Workshop on Combinatorial Algorithms*, volume 10765 of *Lecture Notes in Computer Science*, pages 88–99, 2017.

[100] Hiroki Oshima. Improved randomized algorithm for k-submodular function maximization. *SIAM Journal on Discrete Mathematics*, 35(1):1–22, 2021.

[101] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, volume 32. Courier Corporation, 01 1982.

[102] Canh V. Pham, Dung K. T. Ha, Huan X. Hoang, and Tan D. Tran. Fast streaming algorithms for k-submodular maximization under a knapsack constraint. In *Proceedings of the 9th IEEE International Conference on Data Science and Advanced Analytics*, pages 1–10. IEEE, 2022.

[103] Canh V. Pham, Quang C. Vu, Dung K. Ha, Tai T. Nguyen, and Nguyen D. Le. Maximizing k-submodular functions under budget constraint: applications and streaming algorithms. *Journal of Combinatorial Optimization*, 44(1):723–751, 2022.

[104] Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22(4):595–608, 2018.

[105] Akbar Rafiey and Yuichi Yoshida. Fast and private submodular and k-submodular functions maximization with matroid constraints. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7887–7897. PMLR, 2020.

[106] Yingli Ran, Zhao Zhang, and Shaojie Tang. Improved parallel algorithm for minimum cost submodular cover problem. In *Proceedings of the 35th Annual Conference on Learning Theory*, volume 178, pages 3490–3502. PMLR, 2022.

[107] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 475–484, 1997.

[108] Shinsaku Sakaue. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discrete Optimization*, 23:105–113, 2017.

[109] Alberto Schiabel, Vyacheslav Kungurtsev, and Jakub Marecek. Randomized algorithms for monotone submodular function maximization on the integer lattice. *arXiv preprint arXiv:2111.10175*, 2021.

[110] Alexander Schrijver. A course in combinatorial optimization. *CWI, Kruislaan*, 413:1098, 2003.

[111] Ajit P. Singh, Andrew Guillory, and Jeff A. Bilmes. On bisubmodular maximization. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22, pages 1055–1063. JMLR, 2012.

[112] Tasuku Soma and Yuichi Yoshida. A generalization of submodular cover via the diminishing return property on the integer lattice. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pages 847–855, 2015.

[113] Tasuku Soma and Yuichi Yoshida. Maximizing monotone submodular functions over the integer lattice. *Mathematical Programming*, 172:539–563, 2018.

[114] Xiaoming Sun, Jialin Zhang, Shuo Zhang, and Zhijie Zhang. Improved deterministic algorithms for non-monotone submodular maximization. *preprint*, arXiv:2208.14388, 2022.

[115] Yunjing Sun, Yuezhu Liu, and Min Li. Maximization of k-submodular function with a matroid constraint. In *Proceedings of the 17th Annual Conference on Theory and Applications of Models of Computation*, pages 1–10. Springer, 2022.

[116] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[117] Jingjing Tan, Dongmei Zhang, Hongyang Zhang, and Zhenning Zhang. Streaming algorithms for monotone dr-submodular maximization under a knapsack constraint on the integer lattice. In *Proceedings of the 12th International Symposium on Parallel Architectures, Algorithms and Programming*, pages 58–67. Springer, 2020.

[118] Zhongzheng Tang, Chenhao Wang, and Hau Chan. On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters*, 50(1):28–31, 2022.

[119] Françoise Tisseur and Jack Dongarra. A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures. *SIAM Journal on Scientific Computing*, 20(6):2223–2236, 1999.

[120] Sebastian Tschiatschek, Rishabh Iyer, Haochen Wei, and Jeff Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, page 1413–1421. MIT Press, 2014.

[121] Peng-Jun Wan, Ding-Zhu Du, Panos Pardalos, and Weili Wu. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Applications*, 45(2):463–474, 2010.

[122] Baoxiang Wang and Huanjian Zhou. Multilinear extension of k-submodular functions. *CoRR*, abs/2107.07103, 2021.

[123] Justin Ward and Stanislav Zivný. Maximizing bisubmodular and k-submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1468–1481. SIAM, 2014.

[124] Gerhard J. Woeginger. Exact algorithms for np-hard problems: A survey. *Combinatorial Optimization – Eureka, You Shrink!*, 2570:185–207, 2003.

[125] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[126] Hao Xiao, Qian Liu, Yang Zhou, and Min Li. Non-monotone k-submodular function maximization with individual size constraints. In *Proceedings of the 11th International Conference on Computational Data and Social Networks*, pages 268–279. Springer, 2022.

[127] Leqian Zheng, Hau Chan, Grigorios Loukides, and Minming Li. Maximizing approximately k-submodular functions. In *Proceedings of the 2021 SIAM International Conference on Data Mining*, pages 414–422. SIAM, 2021.