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: Khoa học máy tính
Mã số: 9480101

TÓM TẮT LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

Hà Nội – 2026

Công trình được hoàn thành tại: Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội

Người hướng dẫn khoa học:

1. PGS. TS Hoàng Xuân Huấn
2. PGS. TS Phạm Văn Cảnh

Phản biện: .....

Phản biện: .....

Phản biện: .....

Luận án sẽ được bảo vệ trước Hội đồng cấp Đại học Quốc gia chấm luận án tiến sĩ họp tại Đại học Công nghệ, ĐHQGHN vào hồi giờ ……. ngày ….. tháng ….. năm 2025

NGHIÊN CỨU SINH	**CÁN BỘ HƯỚNG DẪN**
Trần Đình Tân	**1. PGS. TS. Hoàng Xuân Huấn**
	2. PGS. TS. Phạm Văn Cảnh

XÁC NHẬN CỦA ĐƠN VỊ ĐÀO TẠO

Có thể tìm hiểu luận án tại:

- Thư viện Quốc gia Việt Nam
- Trung tâm Thông tin - Thư viện, Đại học Quốc gia Hà Nội

Hà Nội – 2026

Lời mở đầu

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. 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.

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 cận 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 $f : 2^V \to \mathbb{R}$ được gọi là *Submodular* nếu với mọi $A \subseteq B \subseteq V$ và $e \in V \setminus B$, ta có: $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$. Tính chất này phản ánh hiện tượng *lợi ích cận 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.

Do phần lớn các bài toán trong tối ưu tổ hợp là NP-khó , trong trường hợp bài toán có kích thước lớn mà thuật toán chính xác không khả thi và heuristic thiếu bảo đảm, thuật toán xấp xỉ trở thành 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**".

Bảng 0.1: 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	–	✓	✓	✓	–	✓
Lực lượng	✓	✓	✓	✓	✓	? **Bài toán 3**
Chi phí	✓	✓ **Bài toán 1**	✓	✓	✓	?
Matroid	✓	✓			?	?
Chi phí nhóm	?	?	?	? **Bài toán 2**	?	?
Lớp bài toán phủ submodular						
Lực lượng tối thiểu	✓	✓	✓	?	?	?
Chi phí tối thiểu	✓ **Bài toán 4**	✓	?	?	?	?

✓ 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.*

Trong tối ưu tổ hợp, các bài toán liên quan đến hàm submodular xuất hiện dưới nhiều hình thức, tiêu biểu là các **bài toán tối đa và đối ngẫu của bài toán tối đa – phủ hàm submodular**. Song song với sự phát triển của lý thuyết nền tảng, nhiều mở rộng của hàm submodular đã được đề xuất nhằm mô tả tốt hơn

các mô hình thực tế có cấu trúc phức tạp. Trong đó, k-submodular được dùng cho các bài toán phân hoạch nhiều nhóm, còn DR-submodular cho phép mở rộng miền xác định sang lưới nguyên. Ngoài đặc trưng hàm mục tiêu, bài toán tối ưu còn bị chi phối bởi các ràng buộc đi kèm như chi phí, lực lượng, hoặc cấu trúc, v.v. Sự kết hợp giữa các ràng buộc này với tính đơn điệu (hoặc không) của hàm mục tiêu tạo nên một hệ đa dạng các biến thể. Mặc dù đã có nhiều nỗ lực xây dựng các thuật toán xấp xỉ hiệu quả, vẫn còn không ít trường hợp chưa có lời giải thoả đáng *(xem Bảng 0.1)*. Một số bài toán dù đã được tiếp cận nhưng hệ số xấp xỉ còn thấp hoặc độ phức tạp chưa phù hợp thực tiễn, phản ánh sự thiếu hụt trong hệ thống lý thuyết hiện nay, nhất là với các trường hợp hàm không đơn điệu và đa ràng buộc. Dựa trên ma trận phân loại bài toán theo loại hàm (submodular, k-submodular, DR-submodular), tính đơn điệu và dạng ràng buộc *(Bảng 0.1)*, nghiên cứu sinh tiếp cận vấn đề theo hai hướng bổ trợ: Với các bài toán đã có kết quả trước đó, mục tiêu là cải thiện hiệu năng thông qua tăng hệ số xấp xỉ, giảm độ phức tạp truy vấn, hoặc tối ưu cả hai; với các bài toán chưa được nghiên cứu, trọng tâm là xây dựng phương pháp mới cùng bảo đảm lý thuyết rõ ràng về hiệu năng và tính đúng đắn. Cụ thể, nghiên cứu sinh lựa chọn **bốn bài toán** tiêu biểu để khảo sát và phát triển thuật toán, bao gồm:

- **Bài toán nghiên cứu 1:** Bài toán tối đa hàm submodular với ràng buộc chi phí.
- **Bài toán nghiên cứu 2:** Bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm.
- **Bài toán nghiên cứu 3:** Bài toán tối đa hàm DR-submodular với ràng buộc lực lượng.
- **Bài toán nghiên cứu 4:** Bài toán phủ submodular với chi phí tối thiểu (Đối ngẫu của bài toán tối đa hàm submodular với ràng buộc chi phí).

Việc lựa chọn bốn bài toán nêu trên là hợp lý, bởi chúng đại diện cho sự đa dạng trong không gian các loại hàm mục tiêu (từ submodular cổ điển đến các hàm mở rộng như k-submodular và DR-submodular), cũng như trong các dạng ràng buộc thực tiễn (chi phí, chi phí nhóm, lực lượng). Đồng thời, việc kết hợp giữa hai bài toán đã có hướng nghiên cứu nền tảng và hai bài toán mới chưa được khai thác giúp cân bằng giữa chiều sâu lý thuyết và khả năng đóng góp mới cho cộng đồng nghiên cứu. Cách tiếp cận này hứa hẹn mở ra các hướng ứng dụng tiềm năng trong học máy, khai phá dữ liệu và tối ưu hệ thống mạng. 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.

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: **(1) Về mặt lý thuyết:** Đề xuất các thuật toán xấp xỉ mới cho bốn bài toán nghiên cứu và 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. **(2) Về mặt thực nghiệm:** Triển khai và kiểm chứng các thuật toán được đề xuất trên nhiều bộ dữ liệu thực tế có cấu trúc khác nhau và 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. **(3) 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.

Luận án gồm bốn chương chính và phần kết luận, được bố trí theo trình tự từ lý thuyết nền tảng đến kết quả nghiên cứu cụ thể: **Chương 1** trình bày tổng quan về bài toán tối ưu tổ hợp và vai trò của hàm submodular trong các ứng dụng thực tiễn. **Chương 2** nghiên cứu bài toán tối đa hàm submodular với ràng

buộc chi phí - Bài toán nghiên cứu 1. **Chương 3** nghiên cứu bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm - Bài toán nghiên cứu 2 và bài toán tối đa hàm DR-submodular với ràng buộc lực lượng - Bài toán nghiên cứu 3. **Chương 4** nghiên cứu bài toán phủ submodular với chi phí tối thiểu - Bài toán nghiên cứu 4.

Chương 1 Giới thiệu về tối ưu hàm submodular

1.1. Giới thiệu về hàm submodular

1.1.1. Định nghĩa hàm submodular

Định nghĩa 1.1 (Định nghĩa hàm submodular). Cho một tập hữu hạn V có n phần tử, một hàm tập hợp $f : 2^V \to R$ (một ánh xạ từ không gian các tập con của V sang tập số thực) được gọi là *hàm submodular* nếu: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \subseteq V.$

1.1.2. Biến thể hàm submodular

1.1.2.1. Định nghĩa hàm k-Submodular

Định nghĩa 1.2 (Định nghĩa hàm k-submodular). Cho một tập cơ sở V và một số nguyên dương k, quy ước $[k] = \{1, 2, \ldots, k\}$ và $(k+1)^V = \{(V_1, V_2, \ldots, V_k) \mid V_i \subseteq V,\ \forall i \in [k],\ V_i \cap V_j = \emptyset,\ \forall i \neq j\}$ là họ k tập đôi một không giao nhau, được gọi là k-tập (k-*set*). Hàm $f : (k+1)^V \to \mathbb{R}$ là k-*submodular* khi và chỉ khi với các biến $\mathbf{x}, \mathbf{y} \in (k+1)^V$, ta có:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}),$$

trong đó: $\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, \ldots, X_k \cap Y_k)$, và $\mathbf{x} \sqcup \mathbf{y} = (Z_1, \ldots, Z_k)$, với $Z_i = X_i \cup Y_i \setminus (\bigcup_{j \neq i}(X_j \cup Y_j))$.

1.1.2.2. Định nghĩa hàm DR-submodular

Cho V là một tập cơ sở hữu hạn và hàm tập hợp $f : \mathbb{Z}_+^V \to \mathbb{R}$. 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 vectơ $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, lợi ích cận biên của f khi cộng thêm $\mathbf{y}$ vào $\mathbf{x}$ được ký hiệu như sau: $f(\mathbf{y} \mid \mathbf{x}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{x})$ và ký hiệu $\{\mathbf{x}\}$ là tập hợp các phần tử $e \in V$ sao cho $\mathbf{x}(e) > 0$. Khi đó tính chất *submodular trên lưới nguyên* và hàm *DR-submodular* được phát biểu như sau:

Định nghĩa 1.3 (Tính chất *submodular trên lưới nguyên*). Một hàm $f : \mathbb{Z}_+^V \to \mathbb{R}$ được gọi là *submodular trên lưới nguyên* nếu với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, $f(\mathbf{x}) + f(\mathbf{y}) \ \geq\ f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y})$, trong đó $(\mathbf{x} \vee \mathbf{y})(e) = \max\{\mathbf{x}(e), \mathbf{y}(e)\}$ và $(\mathbf{x} \wedge \mathbf{y})(e) = \min\{\mathbf{x}(e), \mathbf{y}(e)\}$.

Định nghĩa 1.4 (Định nghĩa hàm *DR-submodular*). Hàm f gọi là *diminishing-return submodular* (DR-submodular) nếu với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$ sao cho $\mathbf{x} \leq \mathbf{y}$ và mọi $e \in V$, $f(\mathbf{x} + \mathbf{1}_e) - f(\mathbf{x}) \ \geq\ f(\mathbf{y} + \mathbf{1}_e) - f(\mathbf{y})$. Hệ quả bằng quy nạp: $f\big(k\mathbf{1}_e \mid \mathbf{x}\big) \ \geq\ f\big(k\mathbf{1}_e \mid \mathbf{y}\big) \quad \forall k \in \mathbb{Z}_+,\ \mathbf{x} \leq \mathbf{y},\ e \in V.$

1.2. Các bài toán tiêu biểu tối ưu hàm submodular

1.2.1. Bài toán tối đa hàm submodular

Định nghĩa 1.5 (Tối đa hàm submodular với ràng buộc $\mathcal{I}$). 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 $\mathcal{I} \subseteq 2^V$, bài toán được phát biểu như sau: $\max_{S \in \mathcal{I}} f(S)$.

Định nghĩa 1.6 (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 $f : \mathbb{Z}_+^V \to \mathbb{R}_{\geq 0}$, và một tập khả thi tổng quát $\mathcal{I} \subseteq \mathbb{Z}_+^V$, bài toán được phát biểu như sau: $\max_{\mathbf{x} \in \mathcal{I}} f(\mathbf{x})$.

Định nghĩa 1.7 (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 $i \neq j$.

1.2.2. Bài toán phủ Submodular

Định nghĩa 1.8 (Bài toán phủ submodular)**.** 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: $f(S) \geq T$ và $c(S)$ là nhỏ nhất. Bài toán này được phát biểu như sau: $\min_{S \subseteq V, f(S) \geq T} c(S)$.

Định nghĩa 1.9 (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 1.10 (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.

1.3. Các ràng buộc phổ biến của bài toán tối ưu hàm Submodular

1.3.1. Không ràng buộc

Không ràng buộc (unconstrained) 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. Luận án xem xét bài toán tối đa hàm submodular không đơn điệu không có ràng buộc trong Chương 2 để giải quyết một bài toán con.

1.3.2. Ràng buộc về lực lượng

Ràng buộc lực lượng (cardinality constraint hoặc size constraint) 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. Luận án xem xét ràng buộc lực lượng trong **Bài toán nghiên cứu số 3**, với mục tiêu là tối đa hàm *DR-submodular* với ràng buộc lực lượng. Các nội dung chi tiết sẽ được trình bày trong Chương 3.

1.3.3. Ràng buộc về chi phí

Ràng buộc chi phí (knapsack constraint) 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 $c : V \to \mathbb{R}_{>0}$ 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 nhiều bài toán tối ưu *Submodular*. Cụ thể, **Bài toán nghiên cứu số 1** tập trung vào việc tối đa hàm *submodular* không đơn điệu với ràng buộc chi phí; nội dung chi tiết được trình bày trong Chương 2. Tiếp theo, một biến thể mở rộng của ràng buộc chi phí, gọi là ràng buộc chi phí nhóm, được áp dụng cho hàm *k-submodular* và được khai thác trong **Bài toán nghiên cứu số 2**, nội dung được trình bày trong Chương 3. Ngoài ra, **Bài toán nghiên cứu số 4** nhằm tìm tập hợp phủ hàm *submodular* đạt một ngưỡng giá trị cho trước T với tổng chi phí là nhỏ nhất, được trình bày trong Chương 4.

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 số 1**. Đóng góp chính của Chương 2 là đề xuất ba thuật toán xấp xỉ mới để giải bài toán SMK hiệu quả hơn so với các thuật toán hiện hành:(1) **DLA**: thuật toán tất định đạt hệ số xấp xỉ $1/6 - \epsilon$ với độ phức tạp truy vấn tuyến tính $O(n \log(1/\epsilon)/\epsilon)$. So với thuật toán tất định tốt nhất hiện nay SMKDETACC có cùng hệ số xấp xỉ nhưng độ phức tạp cao hơn $O(n \log(k/\epsilon)/\epsilon)$. (2) **RLA**: thuật toán ngẫu nhiên đạt hệ số xấp xỉ $1/4 - \epsilon$, tương đương với thuật toán tốt nhất hiện nay (SMKRANACC có độ phức tạp truy vấn $O(n \log(k/\epsilon)/\epsilon)$), nhưng cải thiện về mặt tốc độ khi số phần tử khả thi k tăng lớn, nhờ giữ nguyên độ phức tạp truy vấn tuyến tính như DLA. (3) **AST**: thuật toán song song có khả năng tận dụng cấu trúc hệ thống tính toán hiện đại, đạt hệ số xấp xỉ $1/7 - \epsilon$, cải thiện so với thuật toán tốt nhất trước đó có hệ số $1/8 - \epsilon$ trong khi vẫn duy trì độ phức tạp song song tối ưu $O(\log n)$ và độ phức tạp truy vấn $\tilde{O}(nk)$.

2.1. Mô tả bài toán

2.1.1. Định nghĩa bài toán

Định nghĩa 2.1 (Bài toán Tối đa hàm Submodular với ràng buộc chi phí –SMK)**.** Cho một tập cơ sở $V = \{e_1, \ldots, e_n\}$ và một hàm mục tiêu submodular $f : 2^V \mapsto \mathbb{R}_+$ 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 rằng 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.

2.1.2. Nghiên cứu liên quan

Các thuật toán tất định, Gupta(2010) là tác giả đầu tiên đề xuất thuật toán đạt hệ số $1/6$ với $O(n^5)$ truy vấn bằng cách kết hợp thuật toán của Sviridenko 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 đạt hệ số $1/10$ trong $O(n^2 \log(n)/\epsilon)$ truy vấn. Công trình của Li(2018) đưa ra thuật toán đạt hệ số $1/9.5 - \epsilon$ với độ phức tạp $O(nk) \max\{\epsilon^{-1}, \log\log n\}$ cho nhiều lớp ràng buộc. Trong mô hình thuật toán dòng, Cui(2021) đề xuất thuật toán với hệ số $\frac{1}{2.05+\rho_{\mathsf{Alg}}}$, độ phức tạp phụ thuộc vào thuật toán offline Alg và số phần tử k. Ngoài ra,

Han(2021) cũng trình bày một thuật toán tất định khác với hệ số $1/6 - \epsilon$ trong $O(n \log(k/\epsilon)/\epsilon)$ truy vấn. Thuật toán có hệ số tốt nhất hiện nay là $1/4 - \epsilon$ của Sun(2022), tuy nhiên cần $O(n^3 \log(n/\epsilon)/\epsilon)$ truy vấn.

Các thuật toán ngẫu nhiên. Công trình đầu tiên được đề xuất bởi Lee(2010) đạt hệ số xấp xỉ $1/5 - \epsilon$, sau đó được cải thiện còn $1/4 - \epsilon$ bởi Kulik(2013). Nhiều nghiên cứu tiếp theo đã nâng cao hiệu quả xấp xỉ với mức $1/e - \epsilon$. Công trình của Buchbinder(2019) 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(2020) đã đề xuất thuật toán SampleGreedy với hệ số $1/5.83 - \epsilon$ và $O(n \log(n/\epsilon)/\epsilon)$ truy vấn. Sau đó, Amanatidis(2021) tiếp tục phát triển thuật toán song song với hệ số $1/9.465 - \epsilon$ trong $O(n^2 \log^2(n) \log(1/\epsilon)/\epsilon^3)$ truy vấn. Gần đây, Han(2021) giới thiệu thuật toán ngẫu nhiên có tốc độ nhanh nhất với hệ số $1/4 - \epsilon$ và $O(n \log(k/\epsilon)/\epsilon)$ truy vấn.

Các thuật toán song song. Trong mô hình tính toán song song, khái niệm độ phức tạp song song được giới thiệu bởi Balkanski(2018), đại diện cho số vòng tuần tự cần thiết nếu các truy vấn có thể thực hiện song song. Trong bài toán SMK, Ene(2019) trình bày thuật toán với hệ số $1/e - \epsilon$ 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(2021) cải thiện độ phức tạp song song xuống $O(\log n)$ với hệ số $1/9.465 - \epsilon$. Gần đây, Cui(2023) đưa ra một thuật toán song song hiệu quả đạt hệ số $1/8 - \epsilon$ 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 $1/7.83 - \epsilon$ 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.

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	$\frac{1}{6} - \epsilon$	–	$O(n \log k)$	Tất định
SMKSTREAM	$\frac{1}{6} - \epsilon$	–	$O(n \log B)/\epsilon$	Tất định, dòng
DLA (Thuật toán 2)	$\frac{1}{6} - \epsilon$	–	$\mathbf{O(n \log(1/\epsilon)/\epsilon)}$	Tất định
Alg.1 trong Buchbinder(2019)	$\frac{1}{2.6}$	–	$poly(n)$	Tất định
FANTOM	$\frac{1}{10} - \epsilon$	–	$O(n^2 \log(n)/\epsilon)$	Ngẫu nhiên
SAMPLEGREEDY	$\frac{1}{5.83} - \epsilon$	–	$O(n \log(n/\epsilon)/\epsilon)$	Ngẫu nhiên
SMKRANACC	$\frac{1}{4} - \epsilon$	–	$O(n \log k)$	Ngẫu nhiên
RLA (Thuật toán 4)	$\frac{1}{4} - \epsilon$	–	$O(n \log(1/\epsilon)/\epsilon)$	Ngẫu nhiên
Alg.4 trong Ene(2019)	$\frac{1}{e} - \epsilon$	$O(\log^2 n)$	$\tilde{O}(n^2)$	Song song
ParKnapsack	$\frac{1}{9.465} - \epsilon$	$\mathbf{O(\log n)}$	$\tilde{O}(n^2)$	Song song
Alg.3 trong Cui(2023)	$\frac{1}{8} - \epsilon$	$\mathbf{O(\log n)}$	$\tilde{O}(nk)$	Song song
ParSKP2	$\frac{1}{7.83} - \epsilon$	$O(\log^2 n)$	$\tilde{O}(nk)$	Song song
AST (Thuật toán 5)	$\mathbf{\frac{1}{7}} - \epsilon$	$\mathbf{O(\log n)}$	$\tilde{O}(nk)$	Song song

Các nghiên cứu hiện tại về bài toán SMK đã đóng góp nhiều thuật toán có hệ số xấp xỉ hằng số với 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. Tuy nhiên, vẫn tồn tại khoảng cách đáng kể giữa các thuật toán có độ phức tạp song song thấp (tức khả năng song song cao) và các thuật toán đạt hệ số xấp xỉ tốt nhất. Hầu hết các thuật toán hiệu quả về độ phức tạp song song hiện nay vẫn yêu cầu truy vấn ở mức gần đa thức, khiến chúng

khó áp dụng trong các ứng dụng thực tế với hạn chế tài nguyên. Bên cạnh đó, một số thuật toán gần đây vẫn tồn tại hạn chế trong việc xử lý hàm mục tiêu không đơn điệu hoặc không đảm bảo hiệu quả khi triển khai trên nền tảng song song. Những hạn chế này mở ra hướng nghiên cứu mới cho việc phát triển thuật toán vừa đạt được hệ số xấp xỉ gần tối ưu, vừa đảm bảo độ phức tạp truy vấn và độ phức tạp song song thấp, hướng tới khả năng ứng dụng rộng rãi trong thực tiễn.

2.2. Thuật toán tất định

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 $B/2$, 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 nghiệm khả thi trên V_2 chỉ chứa tối đa một phần tử, thuật toán ước lượng giá trị tối ưu bởi phần tử đơn lẻ tốt nhất $e_{max} = \arg\max_{e \in V} f(e)$. Đối với V_1, thuật toán khởi tạo hai tập rời nhau X và Y, mỗi tập có một ngưỡng để xét việc thêm phần tử mới dựa trên tỷ lệ giữa lợi ích cận biên và chi phí (tức là *mật độ lợi ích*). Một phần tử được thêm vào tập $Z \in \{X, Y\}$ nếu nó có mật độ lợi ích lớn hơn ngưỡng $f(Z)/B$. Vì tổng chi phí của X và Y có thể vượt quá B, thuật toán chỉ giữ lại phần tử cuối cùng được thêm vào mỗi tập mà vẫn gần nhất với giới hạn chi phí B (dòng 5,6). Thuật toán trả về nghiệm khả thi có giá trị f lớn nhất trong ba tập: phần tử đơn e_{max}, X, hoặc Y sau khi cắt. Cụ thể các bước của thuật toán được trình bày tại Thuật toán 1. Không giống như phương pháp trong Li(2022), 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 dưới đây: $f(O_1) \leq f(X \cup O_1) + f(Y \cup O_1)$, và từ đó suy ra $f(O) \leq f(O_1) + f(O_2)$, trong đó O_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.

Algorithm 1 LA Algorithm

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

1: $V_1 \leftarrow \{e \in V : c(e) \leq B/2\}, X \leftarrow \emptyset, Y \leftarrow \emptyset, e_{max} \leftarrow \arg\max_{e \in V} f(e)$
2: **foreach** $e \in V_1$ **do**
3: Find $Z \in \{X, Y\}$ such that: $Z = \arg\max_{Z \in \{X,Y\}: \frac{f(e|Z)}{c(e)} \geq \frac{f(Z)}{B}} \frac{f(e|Z)}{c(e)}$
4: **If** exist such set Z **then** $Z \leftarrow Z \cup \{e\}$
5: $X' \leftarrow \arg\max_{X(j):0 \leq j \leq |X|, c(X(j)) \leq B} c(X(j))$
6: $Y' \leftarrow \arg\max_{Y(j):0 \leq j \leq |Y|, c(Y(j)) \leq B} c(Y(j))$, where $T(j)$ is a set of last j elements added in $T \in \{X, Y\}$.
7: $S \leftarrow \arg\max_{Z \in \{X', Y', \{e_{max}\}\}} f(Z)$
8: **return** S.

Algorithm 2 DLA Algorithm

Input: An instance (f, V, B), ϵ

1: $S' \leftarrow LA(f, V, B), \Gamma \leftarrow f(S'), \epsilon' \leftarrow \frac{\epsilon}{14}$
2: $\Delta \leftarrow \lceil \frac{\log(1/\epsilon')}{\epsilon'} \rceil, \theta \leftarrow 19\Gamma/(6\epsilon' B), X \leftarrow \emptyset, Y \leftarrow \emptyset$
3: **while** $\theta \geq \Gamma(1 - \epsilon')/(6B)$ **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: $Y'_{(l)} \leftarrow \arg\max_{Y^i : c(Y^i) \leq \epsilon' B (1+\epsilon')^l, i \leq |X|} i$
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: $S \leftarrow \arg\max_{T \in \{S', X, Y, X_{(0)}, .., X_{(\Delta)}, Y_{(0)}, .., Y_{(\Delta)}\}} f(T)$
15: **return** S.

Bổ đề 2.1 cung cấp một giới hạn trên của nghiệm tối ưu 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.

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

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

2.2.2. Thuật toán DLA

Thuật toán DLA (Thuật toán 2) là một thuật toán xấp xỉ tất định với độ phức tạp truy vấn tuyến tính, đạt được tỉ lệ xấp xỉ $1/6 - \epsilon$. Chiến lược chính của DLA là kết hợp các thuộc tính của hai tập rời nhau với kỹ thuật tham lam có ngưỡng để xây dựng nhiều nghiệm ứng viên, từ đó phân tích được tính chất của hàm mục tiêu không đơn điệu. Thuật toán DLA nhận đầu vào là một bài toán (f, V, B) cùng tham số ϵ. Thuật toán gồm hai pha chính. **Trong pha đầu tiên** (dòng 1–7), thuật toán gọi LA như một thủ tục con để thu được một nghiệm ứng viên S' và một khoảng xấp xỉ cho giá trị tối ưu $[\Gamma, 19\Gamma]$, với $\Gamma = f(S')$ (dòng 1). Sau đó, thuật toán áp dụng kỹ thuật tham lam có ngưỡng để thêm các phần tử có mật độ lợi ích cao vào hai tập rời nhau X và Y. Cụ thể, pha này bao gồm nhiều vòng lặp, mỗi vòng duyệt toàn bộ tập cơ sở một lần (dòng 3–7). Một phần tử sẽ được thêm vào tập $T \in \{X, Y\}$ có mật độ lợi ích cao hơn (và không vượt quá ràng buộc chi phí), miễn là mật độ lợi ích đó không nhỏ hơn ngưỡng θ. Ngưỡng này được khởi tạo là $19\Gamma/(6\epsilon')$ và giảm dần theo hệ số $(1 - \epsilon')$ sau mỗi vòng lặp, cho đến khi nhỏ hơn $\Gamma(1 - \epsilon')/(6B)$, với $\epsilon' = \epsilon/14$. **Pha thứ hai** (dòng 8–13) nhằm cải thiện chất lượng của nghiệm ứng viên $T \in \{X, Y\}$ được tạo ra ở cuối pha 1. Gọi T^i là tập gồm i phần tử đầu tiên được thêm vào T trong pha 1. Quan sát chính trong phân tích là hiệu quả của thuật toán phụ thuộc vào chi phí của $T' = \arg\max_{T^i : c(T^i) \leq B - c(r)}(i)$, trong đó $r = \arg\max_{o \in O} c(o)$ và luôn thoả $c(r) \leq B$. Thuật toán thực hiện quét các giá trị khả dĩ cho $c(T')$ từ $\epsilon' B$ đến B và nâng cao chất lượng của T' bằng cách thêm vào phần tử $e = \arg\max_{e \in V : c(T' \cup \{e\}) \leq B} f(T' \cup \{e\})$ (dòng 11–13). Hai Bổ đề sau đây cung cấp giới hạn dưới cho nghiệm cuối cùng của thuật toán trong hai trường hợp: $c(r) < (1 - \epsilon')B$ và $c(r) \geq (1 - \epsilon')B$.

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

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

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

2.3. Thuật toán ngẫu nhiên

2.3.1. Thuật toán LAR

Thuật toán LAR (Thuật toán 3) là phiên bản ngẫu nhiên hóa của Thuật toán 1. Thuật toán LAR chọn tập V_p từ V_1 bằng cách chọn từng phần tử $e \in V_1$ với xác suất $p > 0$, sau đó xây dựng tập ứng viên S từ V_p

thay vì duy trì hai tập rời nhau như trước. 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.

Algorithm 3 LAR Algorithm

Input: An instance (f, V, B), parameters p, α

$e_{max} \leftarrow \max_{e \in V} f(e), V_1 \leftarrow \{e \in V | c(e) \leq B/2\}$
$V_p \leftarrow \{e \in V_1 : \text{Select } e \text{ with probability } p\}, S \leftarrow \emptyset$
foreach $e \in V_p$ **do**
  If $f(e|S)/c(e) \geq \alpha f(S)/B$ **then** $S \leftarrow S \cup \{e\}$
$S' \leftarrow \arg\max_{S(j):0 \leq j \leq |X|, c(X(j)) \leq B} c(S(j))$, where $S(j)$ is a set of last j elements added into S.
$S \leftarrow \arg\max_{T \in \{S', \{e_{max}\}\}} f(T)$
return S.

Algorithm 4 RLA Algorithm

Input: An instance (f, V, B), ϵ

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

Luận án phân tích lý thuyết cho thuật toán LAR qua các bổ đề và định lý sau:

Bổ đề 2.4 (Bổ đề 2.2 trong Feige(2011))**.** *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 $\mathbb{E}[f(A(p))] \geq (1 - p)f(\emptyset) + p \cdot f(A)$.*

Bổ đề 2.5 (Bổ đề 2.2 trong Buchbinder(2014))**.** *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 $\mathbb{E}[f(A(p))] \geq (1 - p)f(\emptyset)$.*

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

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

2.3.2. Thuật toán RLA

Thuật toán RLA (Thuật toán 4) là một thuật toán xấp xỉ ngẫu nhiên với độ phức tạp truy vấn tuyến tính), đạt được tỉ lệ xấp xỉ $1/4 - \epsilon$. Thuật toán RLA kế thừa khung thuật toán của DLA và thực hiện một số điều chỉnh. Cụ thể, thuật toán kết hợp phương pháp tham lam có ngưỡng với một quá trình ngẫu nhiên để xây dựng dãy các nghiệm ứng viên S_j.

Pha đầu tiên của thuật toán bao gồm một vòng lặp (dòng 3–9) thực hiện tối đa $\lceil \log(4/\epsilon')/\epsilon' \rceil$ vòng lặp, mỗi vòng lặp duyệt qua toàn bộ tập cơ sở một lần, trong đó $\epsilon' = \epsilon/10$. Vòng lặp này đồng thời xây dựng một *tập ứng viên* $U = u_1, \ldots, u_j$ và một nghiệm S_j như sau: mỗi phần tử e không thuộc tập ứng viên hiện tại, nếu có mật độ lợi ích không nhỏ hơn ngưỡng θ, sẽ được thêm vào tập U và tiếp tục được đưa vào S_{j+1} với xác suất $1/2$. Tập U đóng vai trò quan trọng đối với hiệu quả của thuật toán RLA.

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).

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 một số ký hiệu sau: Với mỗi $u_i \in U = \{u_1, u_2, \ldots, u_j\}$, định nghĩa $\tau(u_i) = i$, $S^{<u_i} = S_{i-1}$; với mọi $e \in V \setminus U$, đặt $\tau(e) = +\infty$. Ký hiệu $T = j$, nếu $c(S_{j-1} \cup \{u_j\}) \leq B - c(r)$. Ngược lại, $T = \min\{i : 0 \leq i \leq j-1, c(S_i \cup \{u_{i+1}\}) > B - c(r)\}$.
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*

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

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

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

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

2.4. Thuật toán song song

2.4.1. Một số khái niệm liên quan

Để xây dựng được thuật toán AST, luận án phải giải quyết hai bài toán con bên trong: Bài toán tối ưu hàm submodular không ràng buộc và Ngưỡng mật độ. Được phát biểu cụ thể như sau:

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

Định nghĩa 2.3 (Ngưỡng mật độ (**Density Threshold, 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) $\sum_{e \in V \setminus S} f(e|S) \leq \epsilon \cdot \mathsf{opt}$.

Bổ đề 2.8 (Bổ đề 1 trong Cui(2023))**.** *Các tập A, L được sinh bởi RandBatch$(\theta, I, M, \epsilon, f(\cdot), c(\cdot))$ 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 trong Cui(2023))**.** *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.*

Bổ đề 2.10. *Các tập A, L được sinh bởi RandBatch$(\theta, I, M, \epsilon, f(\cdot), c(\cdot))$ 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 5) nhận đầu vào là một bộ ba (f, V, B) cùng các tham số hằng δ, ϵ, α, và bao gồm hai pha chính: **(1)Trong pha đầu tiên**, thuật toán chia tập cơ sở V thành hai tập con: V_0 gồm các phần tử có chi phí nhỏ và V_1 là phần còn lại. Sau đó, thuật toán gọi thủ tục phụ ParSKP1 từ công trình Cui(2023) để thu được một nghiệm xấp xỉ $(1/8 - \delta)$ trong $O(\log n)$ vòng song song. Dựa trên nghiệm này, thuật toán tạo ra $O(\log(1/\epsilon)/\epsilon)$ giá trị dự đoán của nghiệm tối ưu, mỗi giá trị được sử dụng trong một vòng lặp chính. Trong mỗi vòng lặp, thuật toán xây dựng luân phiên hai tập lời giải rời nhau X và Y: X được cập nhật ở các vòng lẻ và Y ở các vòng chẵn. Tại mỗi bước, thuật toán thiết lập một ngưỡng θ_X hoặc θ_Y, gọi thủ tục RandBatch để sinh tập con mới A_i hoặc B_i, rồi cập nhật lời giải và tập còn lại. **(2) Pha thứ hai** nhằm tăng chất lượng lời giải. Nếu $c(X_1 \cup V_0) \leq \epsilon B$, thuật toán áp dụng thuật toán UnSubMax trên tập này để tạo lời giải ứng viên S_1. Sau đó, thuật toán xét các tập con đầu tiên gồm i phần tử trong X và Y, chọn phần tử tốt nhất không vi phạm ngân sách, và trả về lời giải tốt nhất.

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 như của Amanatidis(2021) và Cui(2023), vốn cập nhật cả hai tập lời giải trong một vòng song song duy nhất, và cũng khác với phương pháp *"twin greedy"* bởi Han(2020), 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. Xem xét hai tập X và Y sau khi kết thúc vòng lặp thứ nhất (dòng 4–12). Luận án sử dụng

Algorithm 5 AST Algorithm

Input: An instance (f, V, B), parameters α, ϵ, δ

1: $V_0 \leftarrow \{e \in V : c(e) \leq \epsilon B/n\}$, $V_1 \leftarrow V \setminus V_0$, $I \leftarrow V_1$, $p \leftarrow 1$
2: $S_0 \leftarrow$ ParSKP1$(\frac{1}{4}, \delta, f(\cdot), c(\cdot))$, $\Gamma \leftarrow \frac{8\alpha f(S_0)}{(1-8\delta)\epsilon B}$
3: $X \leftarrow \emptyset, Y \leftarrow \emptyset$, $\Delta \leftarrow \lceil \log_{\frac{1}{1-\epsilon}} \frac{8\alpha}{\epsilon^2(1-8\delta)} \rceil + 1$, $M \leftarrow \frac{1}{\epsilon^2}(\frac{\Delta}{2} + 1)$
4: **for** $i = 1$ *to* Δ **do**
5: **if** i *is odd* **then**
6:   $\theta^X \leftarrow \Gamma(1-\epsilon)^i$
7:   $(A_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^X, I, M, p, \epsilon, f(\cdot|X), c(\cdot))$
8:   $X \leftarrow X \cup A_i$, $I \leftarrow I \setminus X$
9: **else**
10:   $\theta^Y \leftarrow \Gamma(1-\epsilon)^i$
11:   $(B_i, U_i, L_i) \leftarrow$ RandBatch$(\theta^Y, I, M, p, \epsilon, f(\cdot|Y), c(\cdot))$
12:   $Y \leftarrow Y \cup B_i$, $I \leftarrow I \setminus Y$
13: For $T \in \{X, Y\}$, define: T_i is T after the iteration i of the first loop, T^i is the set of first i elements added into T.
14: **if** $c(X_1 \cup V_0) \leq \epsilon B$ **then**
15: $S_1 \leftarrow$ UnSubMax$(X_1 \cup V_0)$
16: **for** $i = 1$ *to* $|X|$ **do**
17: $a_i \leftarrow \arg\max_{e \in V : c(X^i \cup e) \leq B} f(X^i \cup \{e\})$, $X'^i \leftarrow X^i \cup \{a_i\}$
18: **for** $i = 1$ *to* $|Y|$ **do**
19: $b_i \leftarrow \arg\max_{e \in V : c(Y^i \cup e) \leq B} f(Y^i \cup \{e\})$, $Y'^i \leftarrow Y^i \cup \{a_i\}$
20: $S \leftarrow \arg\max_{T \in \{X'^i\}_{i=1}^{|X|} \cup \{Y'^i\}_{i=1}^{|Y|} \cup \{X, Y, S_1\}} f(T)$
21: **return** S

một số ký hiệu sau để trình bày các bổ đề và định lý:

- 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 là nghiệm tối ưu của bài toán SMK trên bộ dữ liệu con (f, V_1, B).

- $O' = O_1 \setminus X_1$, $O^r = O_1 \setminus \{r\}$, và $O'^r = O' \setminus \{r\}$, với r là phần tử có chi phí lớn nhất trong O_1.

- Với một phần tử $e \in X \cup Y$, ký hiệu $X_{<e}$, $Y_{<e}$, θ_e^X, θ_e^Y tương ứng là trạng thái của X, Y, ngưỡng θ^X, θ^Y ngay trước thời điểm e được thêm vào X hoặc Y. Ký hiệu $l(e)$ chỉ số vòng lặp tại đó phần tử e được chọn.

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

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

Định lý 2.5. *Với $\alpha = \frac{1}{7}$, $\epsilon \in (0, \frac{1}{7})$, $\delta \in (0, \frac{1}{8})$, Thuật toán 5 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 $\mathbb{E}[f(S)] \geq (1/7 - \epsilon)\mathsf{opt}$.*

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

Chương này, luận án trình bày hai hướng nghiên cứu chính về hai lớp hàm k-submodular và DR-submodular dưới các ràng buộc lực lượng và chi phí:

Thứ nhất, luận án khảo sát bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm – **Bài toán nghiên cứu số 2**. Để giải bài toán này trong môi trường dữ liệu lớn, luận án đề xuất một thuật toán luồng 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 $O(nk \log(B)/\epsilon)$ và bộ nhớ $O(B \log(n)/\epsilon)$, trong đó B là tổng chi phí và $\epsilon \in (0, 1)$ làm tham số chính xác. Kết quả thực nghiệm cho thấy thuật toán này vượt trội hơn so với các phương pháp tốt nhất hiện nay trong cả chất lượng nghiệm và số lượng truy vấn.

Thứ hai, chương này tập trung vào bài toán tối đa hàm DR-submodular với ràng buộc lực lượng – **Bài toán nghiên cứu số 3**. 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 $O(n \log(k))$, đánh dấu lần đầu tiên có thuật toán xấp xỉ hằng số cho bài toán này với chi phí tính toán thấp. Cả hai thuật toán được kiểm chứng hiệu quả qua các bài toán thực nghiệm tối đa doanh thu với hàm mục tiêu dạng DR-submodular, cho thấy hiệu năng vượt trội cả về tốc độ và chất lượng so với các phương pháp hiện tại.

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

3.1.1. Định nghĩa bài toán

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 của các k-tập $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ với $S_i \subseteq V$, $S_i \cap S_j = \emptyset$ nếu $i \neq j$. 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 rằng 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: $\arg\max_{\mathbf{s} \in (k+1)^V : c_i(\mathbf{s}) \geq B_i, \forall i \in [k]} f(\mathbf{s})$.

3.1.2. Nghiên cứu liên quan

Tối ưu hàm k-submodular. Bài toán tối ưu k-submodular được khởi nguồn từ công trình của Ajit(2012), 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 rằng bài toán k-submodular cũng thuộc lớp bài toán NP-khó. Đối với trường hợp không ràng buộc, Ward(2014) đề xuất một thuật toán tham lam mang tính quyết định với hệ số xấp xỉ $1/3$. Tiếp theo, Iwata(2016) cải tiến bằng một phiên bản tham lam ngẫu nhiên, đạt hệ số xấp xỉ $k/(2k-1)$ thông qua việc kết hợp phân phối xác suất với cách lựa chọn các phần tử có lợi ích cận biên lớn. Sau đó, Hiroki(2017) 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. Naoto(2015) tập trung vào bài toán tối đa hàm k-submodular đơn điệu với ràng buộc lực lượng, và đề xuất một thuật toán tham lam đạt hệ số xấp xỉ $1/2$ trong trường hợp ràng buộc tổng lực lượng, và $1/3$ trong trường hợp đơn lực lượng. Chao(2018) giới thiệu một thuật toán tiến hóa đa mục tiêu cho bài toán tối đa k-submodular đơn điệu với ràng buộc tổng kích thước. Trong khi đó, Leqian(2021) 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. Shinsaku(2017) trình bày một thuật toán tham lam với hệ số xấp xỉ $1/2$ cho bài toán có ràng buộc cấu trúc. Tiếp đó, Rafiey(2020) đề xuất một phương pháp tham lam liên tục riêng biệt, cũng đạt hệ số xấp xỉ $1/2$ cho cùng bài toán. Gần đây, nhiều nghiên cứu đã tập trung vào bài toán k-submodular với ràng buộc chi phí 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, Ward(2014) và Iwata(2016) đã phát triển các thuật toán luồng một lượt quét. Trong đó, Iwata và cộng sự(2016) 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 (2020) là những người đầu tiên đề xuất thuật toán luồng cho bài toán tối đa k-submodular với ràng buộc tổng kích thước trong điều kiện có nhiễu. Họ đưa ra hai thuật toán với hệ số xấp xỉ $O(\epsilon(1-\epsilon)^{-2}B)$ cho hàm đơn điệu và $O(\epsilon(1-\epsilon)^{-3}B)$ cho hàm không đơn điệu. Gần đây, Canh và cộng sự(2022) đề xuất hai thuật toán luồng một lượt quét cho bài toán tối đa k-submodular với ràng buộc chi phí, với độ phức tạp $O(nk\log(n)/\epsilon)$ và hệ số xấp xỉ kỳ vọng lần lượt là $1/4 - \epsilon$ và $k/(4k-1) - \epsilon$.

Theo tìm hiểu của nghiên cứu sinh, hiện chưa có công trình nào nghiên cứu cụ thể bài toán tối đa hàm k-submodular với ràng buộc chi phí nhóm. 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

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 cận 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 $\alpha, \epsilon \in (0, 1)$. Quá trình xử lý dữ liệu được thực hiện trong một lượt quét duy nhất (single-pass), 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 cận 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 cận biên đủ lớn so với ngưỡng $\alpha v/B$ (trong đó $B = \sum_{i=1}^{k} B_i$), thì cặp (e, i_e) sẽ được thêm vào tập ứng viên **s**. Sau khi kết thúc quá trình quét, thuật toán trả về nghiệm tốt nhất giữa tập ứng viên **s** và phần tử đơn $(e_{\max}, i_{\max})$. Chi tiết các bước của thuật toán được trình bày trong Thuật toán 6. 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 trình bày như sau:

- **o** là nghiệm tối ưu của bài toán trên tập V, với giá trị tối ưu tương ứng là $\mathsf{opt} = f(\mathbf{o})$.
- t là số lượng cặp phần tử được thêm vào tập **s** sau khi kết thúc vòng lặp chính.
- $\mathbf{s}^t = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: k-tập **s** sau khi vòng lặp chính kết thúc, với $t = |supp(\mathbf{s}^t)|$.
- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: k-tập **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 $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$.
- v là giá trị dự đoán của opt, thoả mãn: $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$.

Bổ đề 3.1. *Với mọi j, $0 \leq j \leq t$, ta có: (1) Nếu f là hàm đơn điệu, thì $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq f(\boldsymbol{s}^j)$; (2) Nếu f là hàm không đơn điệu, thì $f(\boldsymbol{o}) - f(\boldsymbol{o}^j) \leq 2f(\boldsymbol{s}^j)$.*

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 $\Delta_{(e,i)} f(\boldsymbol{s}^t) < c(e)\alpha v/B_i$, thì: (1) Nếu f là hàm đơn điệu, ta có $f(\boldsymbol{s}^t) > v(1-\alpha k)/2$; (2) Nếu f là hàm không đơn điệu, ta có $f(\boldsymbol{s}^t) > v(1-\alpha k)/3$.*

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 $\Delta_{(e,i)} f(\boldsymbol{s}^t) > c(e)\alpha v/B_i$, thì $f(\boldsymbol{s}) \geq \alpha v/2$.*

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 6 có độ phức tạp truy vấn là $O(nk)$ và: (1) Nếu f là hàm đơn điệu và $\alpha = 1/(k+1)$, thì đạt được tỉ lệ xấp xỉ $(1-\epsilon)/(2(k+1))$; (2) Nếu f là hàm không đơn điệu và $\alpha = 2/(2k+3)$, thì đạt được tỉ lệ xấp xỉ $(1-\epsilon)/(2k+3)$.*

Algorithm 6 STROPT Algorithm

Input: V, f, k, $B_1, B_2, \ldots, B_k$, an approximate value v of opt such that $(1-\epsilon)\mathsf{opt} \leq v \leq \mathsf{opt}$, parameters $\alpha, \epsilon \in (0,1)$.
Output: solution $\mathbf{s}$.
$\mathbf{s} \leftarrow \emptyset$, $(e_{max}, i_{max}) \leftarrow (null, 0)$
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 $\mathbf{s}$

Algorithm 7 STR algorithm

1: **Input:** V, f, k, α, $B_1, B_2, \ldots, B_k (\forall i \in [k])$.
2: **Output:** A solution $\mathbf{s}$
3: $M \leftarrow 0$, $(e_{max}, i_{max}) \leftarrow (null, 0)$
4: **for all** $e \in V$ **do**
5: $i_e \leftarrow \arg\max_{i \in [k]} f((e,i))$
6: $(e_{max}, i_{max}) \leftarrow \arg\max_{\mathbf{x} \in \{(e, i_e), (e_{max}, i_{max})\}} f(\mathbf{x})$
7: $M \leftarrow f((e_{max}, i_{max}))$
8: $O \leftarrow \{v = (1+\epsilon)^j : M \leq (1+\epsilon)^j \leq BM\}$
9: **for all** $v \in O$ **do**
10: $i_v \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v) + c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
11: **if** $\Delta_{(e,i_v)} f(\mathbf{s}_v) \geq c(e)\alpha v / B_{i_e}$ **then**
12: $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
13: **end if**
14: **end for**
15: **end for**
16: **for all** $e \in V \setminus supp(\mathbf{s}_v)$ **do**
17: **for all** $v \in O$ **do**
18: $i_e \leftarrow \arg\max_{i \in [k], c_i(\mathbf{s}_v) + c(e) \leq B_i} \Delta_{(e,i)} f(\mathbf{s}_v)$
19: **if** $\Delta_{(e,i)} f(\mathbf{s}_v) > 0$ **then**
20: $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_e)$
21: **end if**
22: **end for**
23: **end for**
24: $\mathbf{s} \leftarrow \arg\max_{\mathbf{x} \in \{(e_{max}, i_{max})\} \cup \{\mathbf{s}_v : v \in O\}} f(\mathbf{x})$
25: **return** $\mathbf{s}$

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 7), 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: **(1)** 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 tái tạo tương ứng. **(2)** Thêm phần tử e vào các nghiệm ứng viên $\mathbf{s}_v$ với $v \in O$, nếu phần tử này có độ lợi 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 độ lợi biên dương, thuật toán sẽ thêm vào nghiệm ứng viên $\mathbf{s}_v$ tương ứng (dòng 19–20). Chi tiết các bước có thể theo dõi ở mã giả của Thuật toán 7. Luận án phân tích đảm bảo lý thuyết của Thuật toán 7 qua Định lý 3.2.

Định lý 3.2. *Thuật toán 7 thực hiện hai lượt quét qua tập cơ sở, có độ phức tạp truy vấn $O(nk\log(B)/\epsilon)$ và yêu cầu bộ nhớ $O(B\log(B)/\epsilon)$. Về hệ số xấp xỉ: (1) Nếu f là hàm đơn điệu và $\alpha = 1/(k+1)$, thì đạt được tỉ lệ xấp xỉ $(1-\epsilon)/(2(k+1))$; (2) Nếu f là hàm không đơn điệu và $\alpha = 2/(2k+3)$, thì đạt được tỉ lệ xấp xỉ $(1-\epsilon)/(2k+3)$.*

3.2. Bài toán tối đa hàm DR-submodular với ràng buộc lực lượng

3.2.1. Định nghĩa bài toán

Định nghĩa 3.2 (Bài toán tối đa DR-submodular với ràng buộc kích thước – DrSMC)**.** Cho một hàm DR-submodular $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, một lưới nguyên bị chặn bởi $\mathbb{B}$, 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 $\|\mathbf{x}\|_1 = \sum_{e\in V} \mathbf{x}(e)$.

Gần đây, Ene và Nguyen 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)*: $\max_{S\subseteq V'} \quad g(S)$ với điều kiện: $c(S) \leq k$, trong đó $V' = \bigcup_{e\in V}\{(e, j) : j \in [t_e]\}$ 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 $a_{e,j} \leq \varepsilon k$. 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** đạt tỉ lệ xấp xỉ tốt nhất là $(1/4 - \epsilon)$. 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 3.1). 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.

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 không đơn điệu với ràng buộc kích thước. 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 kích thước 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

Hàm DR-submodular đơn điệu. Soma và Yoshida(2015) 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(2018) 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. Lai và cộng sự (2019) đã giới thiệu một thuật toán ngẫu nhiên. Gần đây hơn, Schiabel và cộng sự (2025) 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.

Hàm DR-submodular không đơn điệu. Gottschalk và Peis (2015) 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) 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) 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, 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 điểm qua 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(2010), 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. Trong số đó, phương pháp trong do Buchbinder(2019) đề xuất 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 do Pham(2022) đề xuất 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 3.1: Bảng so sánh các thuật toán cho bài toán DrSMC.

Thuật toán	**Tỉ lệ xấp xỉ**	**Độ phức tạp**	**Phân loại**
RLA +Reduction	$\frac{1}{4} - \epsilon$	$O\left(\frac{n}{\epsilon}\log(k)\log(\frac{1}{\epsilon})\right)$	Ngẫu nhiên
SMKRANACC+Reduction	$\frac{1}{4} - \epsilon$	$O\left(\frac{n}{\epsilon}\log(k)\log(\frac{k}{\epsilon})\right)$	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 kích thước, 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, trong đó kích thước tập cơ sở được tăng lên thành $n(2\log k + 1)$.

3.2.3. Thuật toán đề xuất

3.2.3.1. Thuật toán FastDrSub

Thuật toán FastDrSub(Thuật toán 8) nhận vào một hàm DR-submodular f, một tập cơ sở hữu hạn V có kích thước n, một ngân sách kích thước $k \in \mathbb{Z}_{>0}$, và một tham số ngưỡng $\alpha \in (0, 1)$. Mục tiêu của thuật toán là tìm một vector nghiệm $\mathbf{z} \in \mathbb{Z}^V_{\geq 0}$ 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. 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 $d_{\max}\mathbf{1}_{e_{\max}}$. Tổng thể, thuật toán chỉ yêu cầu $O(n\log k)$ lần truy vấn hàm mục tiêu và bảo đảm một tỉ lệ xấp xỉ hằng số đã được chứng minh. Luận án trình bày kết quả lý thuyết chính của thuật toán FastDrSub trong Định lý 3.3 dưới đây:

Bổ đề 3.4. *Ta có:* $f(\boldsymbol{o}_1 \vee \boldsymbol{x}) + f(\boldsymbol{o}_1 \vee \boldsymbol{y}) \leq 4(f(\boldsymbol{x}) + f(\boldsymbol{y}))$.

Định lý 3.3. *Với một hằng số* $\alpha \in (0, 1)$*, Thuật toán 8 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* $\alpha = \frac{2\sqrt{2}-1}{7}$.

3.2.3.2. Thuật toán FastDrSub+

Thuật toán FastDrSub+(Thuật toán 9) nhận đầu vào là một hàm DR-submodular f, một tập cơ sở E, 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 $\mathbf{s} \in \mathbb{Z}^E_+$ sao

cho $\|\mathbf{s}\|_1 \leq k$, với giá trị $f(\mathbf{s})$ xấp xỉ giá trị tối ưu của bài toán. 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 $\theta = \Gamma/(4k\beta)$. Tiếp theo, thuật toán xây dựng ba vector ứng viên **x**, **y**, và **z** dựa trên chiến lược ngưỡng giảm dần với hệ số giảm bội $(1-\epsilon)$. Trong quá trình này, hai vector **x** và **y** được cập nhật luân phiên nhằm đảm bảo rằng các phần tử được chọn thuộc hai tập rời nhau, trong khi vector **z** được xây dựng một cách độc lập theo hướng tham lam. Ở mỗi vòng lặp, với mọi phần tử $e \in E$, 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 $\epsilon\Gamma/(16k)$, khi đó thuật toán đánh giá hàm mục tiêu f trên bốn ứng viên: $\mathbf{s}'$, **x**, **y**, và **z**, và trả về vector ứng viên có giá trị hàm mục tiêu lớn nhất. Luận án phân tích và trình bày đảm bảo lý thuyết của thuật toán FastDrSub+ trong Định lý 3.4:

Algorithm 8 FastDrSub Algorithm

Input: $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+, V, k, \alpha$

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**

Algorithm 9 Fast Approximation Plus Algorithm (FastDrSub+)

Input: $f : \mathbb{Z}_+^V \to \mathbb{R}_+, V, k, \alpha, \beta, \epsilon$

1: $\mathbf{s}' \leftarrow$ FastDrSub(f, V, k, α)

2: $\Gamma = f(\mathbf{s}')\left(8\frac{(2-\alpha)}{1-\alpha} + \frac{1}{\alpha}\right),\ \theta = \frac{\Gamma}{4k\beta}$

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: $\quad$ **foreach** $e \in V$ **do**

6: $\quad\quad d_{\mathbf{x}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{x}\|_1,\ f(\mathbf{1}_e \mid \mathbf{x} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search

7: $\quad\quad d_{\mathbf{y}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{y}\|_1,\ f(\mathbf{1}_e \mid \mathbf{y} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search

8: $\quad\quad d_{\mathbf{z}} \leftarrow \max\{d \in \mathbb{Z} \mid 0 < d \leq k - \|\mathbf{z}\|_1,\ f(\mathbf{1}_e \mid \mathbf{z} + (d-1)\mathbf{1}_e) \geq \theta\}$ by using the binary search

9: $\quad\quad \mathbf{z} \leftarrow \mathbf{z} + d_{\mathbf{z}}\mathbf{1}_e$

10: $\quad\quad$ **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: $\quad\quad\quad \mathbf{x} \leftarrow \mathbf{x} + d_{\mathbf{x}}\mathbf{1}_e$

12: $\quad\quad\quad \mathbf{y} \leftarrow \mathbf{y} - \mathbf{y}(e)\mathbf{1}_e$

13: $\quad\quad$ **else**

14: $\quad\quad\quad \mathbf{y} \leftarrow \mathbf{y} + d_{\mathbf{y}}\mathbf{1}_e$

15: $\quad\quad\quad \mathbf{x} \leftarrow \mathbf{x} - \mathbf{x}(e)\mathbf{1}_e$

16: $\quad \theta \leftarrow (1-\epsilon)\theta$

17: $\mathbf{s} \leftarrow \arg\max_{\mathbf{t}\in\{\mathbf{s}', \mathbf{x}, \mathbf{y}, \mathbf{z}\}} f(\mathbf{t})$

18: **return s**

Định lý 3.4. *Với $\epsilon > 0$ và hằng số $\alpha \in (0,1)$, Thuật toán 9 đạt được tỷ lệ xấp xỉ là $\frac{1}{4} - \epsilon$ và có độ phức tạp truy vấn là $O\left(\frac{n}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\log(k)\right)$.*

Chương 4 Bài toán phủ submodular với chi phí tối thiểu

Chương này, luận án tập trung vào bài toán phủ submodular chi phí tối thiểu (Minimum cost Submodular Cover –viết tắt là MSC)– **Bài toán nghiên cứu số 4**, một bài toán cơ bản trong tối ưu submodular có nhiều ứng dụng trong học máy và trí tuệ nhân tạo. Đóng góp chính của chương là đề xuất ba thuật toán luồng xấp xỉ hai tiêu chí mới với hiệu quả lý thuyết và thực nghiệm vượt trội so với các phương pháp hiện hành:

- 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 $\rho = \frac{c_{\max}}{c_{\min}}$ biểu thị tỉ lệ giữa chi phí phần tử lớn nhất và chi

phí phần tử nhỏ nhất.

- ThreeStr: thuật toán ba lượt quét đạt cùng hệ số xấp xỉ như SingStr nhưng cải thiện đáng kể độ phức tạp truy vấn và bộ nhớ xuống $O\left(\frac{n}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$ và $O\left(\frac{k}{\epsilon}\log\left(\frac{n}{\epsilon}\right)\right)$. Đây là thuật toán luồng đầu tiên có độ phức tạp đa thức.

- MultiStr: thuật toán nhiều lượt quét hằng số $O\left(\frac{1}{\epsilon}\log\left(\frac{1}{\epsilon}\right)\right)$, đạt gần như hệ số xấp xỉ tốt nhất của thuật toán tham lam $\left(\left(1+\ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon}, 1-\epsilon\right)$, trong khi giảm mạnh số truy vấn và lượt quét.

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 hội nghị quốc tế **CSoNet 2023 (SCOPUS)**.

4.1. Mô tả bài toán

4.1.1. Định nghĩa bài toán

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 chất lượng $T > 0$, mục tiêu là tìm tập $S \subseteq V$ sao cho: $f(S) \geq T$ và $c(S)$ 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 dạng mô-đun (hoặc cộng tính) c, tức là $c(S) = \sum_{e\in S} c(e)$, 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 $\mathsf{opt} = c(O)$.

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 $NP \subset DTIME(n^{O(\log\log n)})$. 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 đề xuất vào năm 1982 và được cải thiện bởi Wan vào năm 2010. Để nâng cao hiệu năng, 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. 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(2018) 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 rằng bất kỳ thuật toán luồng nào chỉ quét qua dữ liệu một lần mà đạt được hệ số xấp xỉ tốt hơn $n/2$ đều phải sử dụng ít nhất $O(n)$ bộ nhớ đối với bài toán MSC. Gần đây hơn, Crawford (2023) đã giới thiệu hai thuật toán, SINGLE và MULTI, cho bài toán MSC trong trường hợp không đơn điệu. Thuật toán MULTI cung cấp tỉ lệ xấp xỉ hai tiêu chí $\left((1+\epsilon)\left(1+\frac{4}{\epsilon^2}\right), \frac{1-\epsilon}{2}\right)$ 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 $O\left(\frac{\log(\mathsf{opt})}{c_{\min}}\right)$ 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 $\Omega\left(n^2/c_{\min}\right)$. Cần nhấn mạnh rằng 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 cho thể xem ở Bảng 4.1.

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	$\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\log(1+\epsilon)}$	$\left((1+\epsilon)(\frac{4}{\epsilon^2}+1), 1-\epsilon\right)$	$O\left(\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\epsilon\log(1+\epsilon)}(n+\frac{\mathsf{opt}}{\epsilon c_{min}})\right)$	$(1+\epsilon)(\frac{4}{\epsilon^2}+1)\mathsf{opt}$
SINGLE	$\frac{\log(\frac{\mathsf{opt}}{c_{min}})}{\log(1+\epsilon)}$	$\left((1+\epsilon)(\frac{4}{\epsilon^2}+1), 1-\epsilon\right)$	$O\left(\frac{nB\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\epsilon^2 c_{min}\log(1+\epsilon)}\right)$	$\frac{B(1+\epsilon)(\frac{4}{\epsilon^2}+1)\log\left(\frac{2B}{\epsilon T\min_{u\in V}\frac{c(u)}{f(u)}}\right)}{\log(1+\epsilon)}$
Greedy	$O(n)$	$\left(1+\ln(\frac{1}{\epsilon}), 1-\epsilon\right)$	$O(n^2)$	$O(k)$

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, được xác định bởi: $k = \max\{|S| : S \subseteq V, f(S) \geq T$ và $\exists e \in S, f(S \setminus \{e\}) < T\}$. Các thuật toán MULTI và SINGLE 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

4.2.1. Thuật toán luồng một lượt quét SingStr

Thuật toán SingStr (Thuật toán 10) 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)$\frac{f(e\,|\,S_u)}{c(e)} \geq \frac{\alpha T}{\beta u}$ và $c(S_u \cup \{e\}) \leq \beta u$. Điều kiện thứ nhất đảm bảo rằng nghiệm ứng viên chưa đạt ngưỡng bao phủ, còn điều kiện thứ hai yêu cầu phần tử e có mật độ lợi ích trên chi phí đủ cao 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). Luận án phân tích lý thuyết và bảo đảm xấp xỉ của thuật toán SingStr:

Định lý 4.1. *Thuật toán 10 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 $f(S) \geq (1-\epsilon)T$ và $c(S) \leq \frac{(1+\epsilon)(2-\epsilon)}{\epsilon}\mathsf{opt}$.*

4.2.2. Thuật toán luồng ba lượt quét ThreeStr

Thuật toán ThreeStr (Thuật toán 11) được phát triển nhằm cải thiện hiệu năng thực tế của SingStr bằng cách giảm đáng kể độ phức tạp truy vấn trong khi vẫn giữ nguyên tỷ lệ xấp xỉ. Thuật toán thực hiện ba lượt quét dữ liệu: **(1) 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:

Algorithm 10 Thuật toán luồng một lượt quét (SingStr)

Input: An instance (V, f, T), $c_{min}, c_{max}, \epsilon > 0$
Output: A solution S

1: $U = \{c_{min}(1+\epsilon)^{v-1} : \frac{c_{min}}{1+\epsilon} \leq c_{min}(1+\epsilon)^{v-1} \leq nc_{max}, v \in \mathbb{N}\}$
2: $S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
3: $\alpha \leftarrow 2(1-\epsilon), \beta \leftarrow \frac{(1+\epsilon)\alpha}{\epsilon}$
4: **foreach** *incomming element* $e \in V$ **do**
5: **foreach** $u \in U$ **do**
6: **if** $f(S_u) < (1-\epsilon)T$ **then**
7: **if** $\frac{f(e|S_u)}{c(e)} \geq \frac{\alpha T}{\beta u}$ *and* $c(S_u \cup \{e\}) \leq \beta u$ **then**
8: $S_u \leftarrow S_u \cup \{e\}$
9: **if** $c(e) \leq (1+\epsilon)u$ **then**
10: $e_u \leftarrow \arg\max_{x \in \{e, e_u\}} f(x)$
11: $S'_u \leftarrow \arg\max_{X \in \{\{e_u\}, S_u\}} f(X), \forall u \in U$
12: $S \leftarrow \arg\min_{X \in \{S'_u : f(S'_u) \geq (1-\epsilon)T, u \in U\}} c(X)$
13: **return** S

$c'_{\min} = \epsilon c(u_j)/n$ và $c'_{\max} = j \cdot c(u_j)$, dùng trong các bước tiếp theo. **(2) 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. **(3) 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 $S \cup V_0$. 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, 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 $c_{\min}$. 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.

Bổ đề 4.1. *Sau khi kết thúc lượt quét đầu tiên của Thuật toán 11, danh sách $L = \{u_1, u_2, \ldots, u_j\}$ thoả mãn:(1) $c(u_j) \leq c(O) \leq c(\{u_1, \ldots, u_j\}) \leq j \cdot c(u_j)$; (2)$|L| \leq k$; (3) $\mathsf{opt} \leq c_{\max}$.*

Định lý 4.2. *Thuật toán 11 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 $f(S) \geq (1-\epsilon)T$ và $c(S) \leq \frac{\mathsf{opt}}{\epsilon}$.*

4.2.3. Thuật toán luồng nhiều lượt quét MultiStr

MultiStr (Thuật toán 12), 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 **?**, nhằm lọc ra các phần tử có mật độ 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 12. Điều kiện này cho phép các phần tử có mật độ giảm dần được chọn tại mỗi vòng lặp, nhờ đó nâng cao chất lượng nghiệm mà vẫn đảm bảo tổng chi phí không vượt ngưỡng đã định. Luận án trình bày kết quả lý thuyết cho thuật toán MultiStr theo các bổ đề và định lý sau:

Algorithm 11 Thuật toán luồng ba lượt quét

Input: An instance (V, f, T), $c_{min}, c_{max}, \epsilon > 0$
Output: A solution S

1: **foreach** *incomming element* e **do**
2: Insert e into L in non-decreasing cost order
3: Let $L = \{u_1, u_2, \ldots, u_l\}$
4: **if** $f(L) \geq T$ **then**
5: Find $j \leftarrow \min\{i : f(\{u_1, u_2, \ldots, u_i\}) \geq T\}$
6: Update: $L \leftarrow \{u_1, \ldots, u_j\}$
7: $c'_{min} \leftarrow \epsilon c(u_j)/n$, $c'_{max} \leftarrow jc(u_j)$
8: **foreach** *incomming element* $e \in V$ **do**
9: **if** $c(e) < c'_{min}$ **then**
10: $V_0 \leftarrow V_0 \cup \{e\}$
11: **else**
12: **if** $c(e) \in [c'_{min}, c'_{max}]$ **then**
13: $C \leftarrow C + c(e)$
14: $T' \leftarrow T - f(V_0)$, Define $f'(\cdot) \leftarrow f(\cdot|V_0)$
15: $U = \{c'_{min}(1+\epsilon)^v : \frac{c'_{min}}{1+\epsilon} \leq c'_{min}(1+\epsilon)^v \leq C, v \in \mathbb{N}\}$
16: $S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
17: $\alpha = 2(1-\epsilon)$,$\beta = \frac{(1+\epsilon)\alpha}{2-\alpha}$
18: **foreach** *incomming element* $e \in V$ **do**
19: **if** $c(e) \in [c'_{min}, c'_{max}]$ **then**
20: **foreach** $u \in U$ **do**
21: **if** $f'(S_u) < (1-\epsilon)T'$ **then**
22: **if** $\frac{f'(e|S_u)}{c(e)} \geq \frac{\alpha T'}{\beta u}$ *and* $c(S_u \cup \{e\}) \leq \beta u$ **then**
23: $S_u \leftarrow S_u \cup \{e\}$
24: **if** $c(e) \leq (1+\epsilon)u$ **then**
25: $e_u \leftarrow \arg\max_{x \in \{e, e_u\}} f'(x)$
26: $S'_u \leftarrow \arg\max_{X \in \{\{e_u\}, S_u\}} f'(X), \forall u \in U$
27: $S \leftarrow \arg\min_{X \in \{S'_u : f'(S'_u) \geq (1-\epsilon)T', u \in U\}} c(X)$
28: **return** $S \cup V_0$.

Algorithm 12 Thuật toán luồng nhiều lượt quét

Input: An instance (V, f, T), $c_{min}, c_{max}, \epsilon > 0$
Output: A solution S

1: **foreach** *incomming element* e **do**
2: Insert e into in non-decreasing cost order
3: Let $L = \{u_1, u_2, \ldots, u_l\}$
4: **if** $f(L) \geq T$ **then**
5: Find $j \leftarrow \min\{i : f(\{u_1, u_2, \ldots, u_i\}) \geq T\}$
6: Update: $L \leftarrow \{u_1, \ldots, u_j\}$
7: $c'_{min} \leftarrow \epsilon c(u_j)/n$, $c'_{max} \leftarrow jc(u_j)$
8: **foreach** *incomming element* $e \in V$ **do**
9: **if** $c(e) < c'_{min}$ **then**
10: $V_0 \leftarrow V_0 \cup \{e\}$
11: **else**
12: **if** $c(e) \in [c'_{min}, c'_{max}]$ **then**
13: $C \leftarrow C + c(e)$
14: $T' \leftarrow T - f(V_0)$, define $f'(\cdot) \leftarrow f(\cdot|V_0)$
15: $U = \{c'_{min}(1+\epsilon)^v : \frac{c'_{min}}{1+\epsilon} \leq c'_{min}(1+\epsilon)^v \leq C, v \in \mathbb{N}\}$
$S_u \leftarrow \emptyset, e_u \leftarrow null, \forall u \in U$
16: Define $g(\cdot) \leftarrow \min\{T', f'(\cdot)\}$
17: **for** $i = 1$ *to* $\frac{1}{\epsilon}\ln(\frac{1}{\epsilon})$ **do**
18: **foreach** *incomming element* $e \in V$ **do**
19: **if** $c(e) \in [c'_{min}, c'_{max}]$ **then**
20: **foreach** $u \in U$ **do**
21: **if** $g(S_u) < (1-\epsilon)T'$ **then**
22: **if** $\frac{g(e|S_u)}{c(e)} \geq (1-\epsilon)\frac{T'-g(S_u)}{u}$ **then**
23: $S_u \leftarrow S_u \cup \{e\}$
24: $S \leftarrow \arg\min_{X \in \{S'_u : f'(S'_u) \geq (1-\epsilon)T', u \in U\}} c(X)$
25: **return** $S \cup V_0$.

Bổ đề 4.2. *Với thể hiện (V', g, T') của bài toán MSC sau dòng 16 của Thuật toán 12, 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 $c(S_u) \leq \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+\epsilon}{1-\epsilon}\mathsf{opt}'$.*

Định lý 4.3. *Thuật toán 12 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 $c(S) \leq \left(1 + \ln\left(\frac{1}{\epsilon}\right)\right)\frac{1+2\epsilon}{1-\epsilon}\mathsf{opt}$.*

Kết luận

Luận án tập trung nghiên cứu các thuật toán xấp xỉ cho bài toán tối ưu hàm submodular và các mở rộng của nó trong bối cảnh có ràng buộc thực tiễn. 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. Dựa trên khoảng trống lý thuyết được phân tích, luận án đã lựa chọn và nghiên cứu bốn bài toán tiêu biểu:

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 $1/6 - \epsilon$, $1/4 - \epsilon$ và $1/7 - \epsilon$. Các thuật toán này không chỉ đạt độ phức tạp truy vấn tuyến tính và song song tối ưu, mà còn thể hiện hiệu quả vượt trội trong các môi trường tính toán tuần tự và song song. Các kết quả đã được công bố tại hai hội nghị quốc tế xếp hạng A* là IJCAI 2023 và IJCAI 2024.

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 là hướng tiếp cận đầu tiên hiệu quả cho bài toán tối ưu k-submodular trong bối cảnh ràng buộc chi phí nhóm. Kết quả nghiên cứu đã được công bố trên hội thảo quốc tế SoICT 2023(SCOPUS) và tạp chí quốc tế APJOR (SCIE, 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. Đây là những thuật toán đầu tiên đạt được xấp xỉ hằng số cho bài toán này, đồng thời thể hiện hiệu quả vượt trội trong bài toán tối đa doanh thu với hàm mục tiêu DR-submodular. Kết quả đã được công bố tại tạp chí JOCO(SCIE, Q2).

4. **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 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 đã được công bố tại hội nghị quốc tế CSoNet 2023.

Từ kết quả đạt được, luận án mở ra nhiều hướng nghiên cứu tiếp theo có tiềm năng ứng dụng và phát triển lý thuyết. Một số hướng đi triển vọng gồm: (i) mở rộng thuật toán sang các bài toán có ràng buộc tổ hợp phức tạp như matroid, công bằng hoặc đa ràng buộc đồng thời; (ii) phát triển thuật toán học tăng cường hoặc học sâu có đảm bảo lý thuyết cho tối ưu submodular; (iii) thiết kế các thuật toán trực tuyến hoặc phân tán để xử lý bài toán trong môi trường động và dữ liệu phân tán; và (iv) áp dụng các thuật toán vào các lĩnh vực như học máy tích, chọn dữ liệu huấn luyện hiệu quả, thiết kế mạng cảm biến hoặc kiểm soát ảnh hưởng xã hội. Những hướng này không chỉ mở rộng kết quả của luận án mà còn góp phần thúc đẩy sự phát triển của lĩnh vực tối ưu tổ hợp trong môi trường dữ liệu hiện đại.

DANH MỤC CÔNG TRÌNH KHOA HỌC LIÊN QUAN ĐẾN LUẬN ÁN

1. *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. *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. *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. *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. *k-Submodular Maximization Under Individual Knapsack Constraints: Applications and Streaming Algorithm*, 2025, Asia-Pacific Journal of Operational Research **(SCIE, Q3)**.

6. *Fast Approximation Algorithm for Non-Monotone DR-submodular Maximization under Size Constraint*, 2025, Journal of Combinatorial Optimization **(SCIE, Q2)** *(Accepted with minor revision)*.