ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Hà Thị Kim Dung

MỘT SỐ THUẬT TOÁN XẤP XỈ CHO BÀI TOÁN TỐI ƯU HÀM DẠNG SUBMODULAR VỚI RÀNG BUỘC

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

Hà Nội – 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Hà Thị Kim Dung

MỘT SỐ THUẬT TOÁN XẤP XỈ CHO BÀI TOÁN TỐI ƯU HÀM DẠNG SUBMODULAR VỚI RÀNG BUỘC

Ngành: Khọc học máy tính
Chuyên Ngành: Khoa học Máy tính
Mã số: 948010101

LUẬN ÁN TIẾN SĨ KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC:

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

Hà Nội – 2024

LỜI CAM ĐOAN

Tôi xin cam đoan luận án này là kết quả nghiên cứu của tôi, được thực hiện dưới sự hướng dẫn của PGS.TS Hoàng Xuân Huấn và TS. Phạm Văn Cảnh. Các kết quả và số liệu trình bày trong luận án là hoàn toàn trung thực và chưa từng được công bố trong bất kỳ công trình của ai khác. Các nội dung trích dẫn từ các nghiên cứu của các tác giả khác mà tôi trình bày trong luận án này đã được ghi rõ nguồn trong phần tài liệu tham khảo.

Hà Nội, ngày tháng năm 2024

Người thực hiện

Hà Thị Kim Dung

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn tới các Thầy, Cô trong Bộ môn Khoa học máy tính cùng các Thầy, Cô của Khoa Công nghệ thông tin, Đại học Công nghệ, Đại học Quốc gia Hà Nội đã nhiệt tình hỗ trợ, hướng dẫn em trong suốt quá trình học tập và nghiên cứu tại trường.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc tới PGS.TS Hoàng Xuân Huấn, TS. Phạm Văn Cảnh đã tận tình hướng dẫn, định hướng, chỉ dẫn cho em hoàn thành luận án này.

Bên cạnh đó, em cũng xin chân thành cảm ơn các đồng nghiệp trong nhóm nghiên cứu đã tích cực đồng hành, hỗ trợ để em có thể hoàn thành luận án này.

Mặc dù đã cố gắng hết sức, nhưng không thể tránh khỏi những sai sót trong quá trình hoàn thành luận án. Kính mong nhận được sự nhận xét, góp ý của các quý Thầy, Cô và các đồng nghiệp để em có thể hoàn thiện tốt hơn nội dung của luận án.

Hà Nội, ngày tháng năm 2024

Người thực hiện

Hà Thị Kim Dung

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
CO	Combinatorial Optimizationn	Tối ưu tổ hợp
DRSC	DR Submodular Cover	Tập phủ trên lưới nguyên
DS	Deterministic Streaming	Luồng tất định
IM	Influence Maximization	Tối đa ảnh hưởng
IC	Independent Cascade	Bậc độc lập
kCMK	k-topic information Coverage Maximization under Knapsack constraint	Tối đa độ phủ thông tin k-chủ đề với ràng buộc chi phí
kIMK	k-topic Influence Maximization under Knapsack constraint	Tối đa ảnh hưởng k-chủ đề với ràng buộc chi phí
kSMK	k-submodular Maximization under Knapsack constraint	Tối đa hàm k-Submodular với ràng buộc chi phí
kSPK	k-Sensor Placement under Knapsack constraint	Đặt k sensor cảm biến với ràng buộc chi phí
LT	Linear Threshold	Ngưỡng tuyến tính
MXH	-	Mạng xã hội
NCS	-	Nghiên cứu sinh
RIS	Revese Influence Sampling	Lấy mẫu ảnh hưởng ngược
RR	Reachable Reverse	Ảnh hưởng ngược
RS	Random Streaming	Luồng ngẫu nhiên
s.t	Subject to	Sao cho
SC	Submodular Cover	Tập phủ Submodular
SM	Submodular Maximization	Tối đa hàm Submodular
SMK	Submodular Maximization under Knapsack constraint	Tối đa hàm Submodular với ràng buộc chi phí
SMKN	Submodular Maximization subject to a Knapsack constraint under Noises	Tối đa hàm Submodular ràng buộc chi phí với nhiễu

MỞ ĐẦU

1. Bối cảnh nghiên cứu của tối ưu tổ hợp hàm dạng submodular

Tối ưu tổ hợp là một công cụ cơ bản được ứng dụng trong nhiều lĩnh vực khoa học, kỹ thuật, y học, kinh tế..., đặc biệt là khoa học máy tính [45, 123].

Trong các bài toán tối ưu tổ hợp, có nhiều bài toán có hàm mục tiêu là một dạng hàm thu thập và xử lý thông tin. Khi đó, yêu cầu đặt ra là cần phải thu thập được càng nhiều thông tin phong phú, đa dạng càng tốt. Những bài toán như vậy thường dẫn đến tìm lời giải *bài toán tối ưu tổ hợp với hàm mục tiêu dạng submodular*, chẳng hạn, các bài toán tóm tắt tài liệu tự động, bài toán trích chọn đặc trưng, phân tích và tiền xử lý dữ liệu, tối đa ảnh hưởng trên mạng xã hội, đặt các cảm biến [98, 90, 18, 78, 43, 60, 81]... Do vậy, chủ đề nghiên cứu về tối đa hàm submodular và các biến thể của nó là một chủ đề nóng, thu hút rất nhiều các nhà khoa học quan tâm nghiên cứu và công bố tại các Hội nghị hàng đầu về trí tuệ nhân tạo và học máy như IJCAI [39, 120], AAAI [86], NEURIPS [7, 2], ICML[96, 97, 87, 40], SODA [9], STOC [11, 10]... cũng như tại các tạp chí nổi tiếng về học thuật về tối ưu hoá, vận trù học và tối ưu tổ hợp [101, 145, 3, 131, 129]... trong khoảng 20 năm nay, đặc biệt trong thời gian gần đây [86, 85, 34, 1, 67, 24, 41, 35, 44]...

Một cách tóm tắt, hàm submodular là hàm có tính chất *lợi nhuận hiệu suất giảm dần (disminishing return property)*, có nghĩa là, cho tập dữ liệu cơ sở V có kích thước n, hàm mục tiêu $f : 2^V \mapsto \mathbb{R}_+$ là hàm tập hợp, thỏa mãn tính chất: $\forall A \subseteq B \subseteq V, f(A \cup \{e\}) \geq f(B \cup \{e\}), \forall e \in V \setminus B$. Tính chất lợi nhuận hiệu suất giảm dần minh họa rằng giá trị đóng góp của một phần tử vào nhóm nhỏ hơn sẽ có giá trị hơn giá trị đóng góp của phần tử đó vào nhóm lớn hơn. Ví dụ khi cửa hàng chưa bán được sản phẩm nào, lợi nhuận bán một mặt hàng là 100.000đ sẽ có giá trị đóng góp to lớn hơn khi cửa hàng đó đã thu về hàng trăm hay hàng triệu đồng tiền lãi.

Bài toán tối đa hàm submodular yêu cầu tìm lời giải $S \subseteq V$ sao cho hàm $f(\cdot)$ submodular đã cho đạt giá trị cực đại. Trong đó, việc xem xét tìm lời giải sao cho tối đa hàm mục tiêu trong điều kiện có ràng buộc được quan tâm hơn cả. Lý do vì các ràng buộc được đưa vào khiến cho bài toán gần gũi với các yêu cầu thực tế như nguồn nhân lực, tiền của, thời gian... luôn bị hạn chế. Từ đó, hình thành một lớp các bài toán như tối đa hàm submodular với ràng buộc lực lượng, với ràng buộc matroid, hoặc với ràng buộc chi phí...

Hiện nay, giải các bài toán tối ưu hàm submodular cần các thuật toán nhanh, hiệu quả trở nên vô cùng cấp thiết. Vì dữ liệu đầu vào cần xử lý ngày càng tăng

nhanh, làm cho không gian tìm kiếm lời giải trở nên khổng lồ. Do vậy, việc tìm lời giải chính xác trở nên bất khả thi. Qua phân tích từ các nguồn tài liệu của các Giáo sư đầu ngành như Nemhauser [102, 101], Wolsey [145, 146], A. Krause [80], J. Bilmes [13], C.Borg [16], R. Iyer [120] cùng một số công bố tại các hội nghị và tạp chí chuyên ngành nổi tiếng khác, cho thấy *xu hướng đề xuất các thuật toán xấp xỉ cho lời giải cạnh tranh* với các đảm bảo lý thuyết về độ phức tạp thời gian, độ phức tạp bộ nhớ... đang chiếm ưu thế. Thuật toán xấp xỉ cho thấy ưu thế khi chỉ ra được tỉ lệ xấp xỉ của lời giải so với lời giải tối ưu là bao nhiêu, đồng thời phân tích so sánh được các đảm bảo lý thuyết khác giữa các công bố có liên quan.

Một lý do quan trọng nữa khi đề xuất các thuật toán xấp xỉ giải quyết bài toán tối đa hàm submodular là cách tiếp cận này có tính tổng quát. Các thuật toán đề xuất có thể áp dụng trên nhiều bộ dữ liệu khác nhau. Các thuật toán đều được xây dựng trên giả định dữ liệu đầu vào là một tập cơ sở V có kích thước $n > 0$ bất kỳ, một hàm f submodular đã giả định tồn tại một ước lượng cho nó, các sai số cho phép ($\epsilon, \delta.. \in (0, 1)$). Bài toán đã có mô hình tính toán, thuật toán xây dựng từ nó, các phép suy dẫn... đều đã được chứng minh tính chặt chẽ, đúng đắn bằng các bổ đề, định lý và hệ quả thì thuật toán đó có thể làm việc với mọi bộ dữ liệu có các kích cỡ khác nhau. Thực nghiệm của các thuật toán, do vậy là sự kiểm chứng sự tiệm cận giữa lý thuyết và thực nghiệm.

Xuất phát từ bối cảnh phát triển, cùng với các nhu cầu về phân tích dữ liệu của kỷ nguyên số, NCS và cộng sự đã tập trung nghiên cứu đề tài: "***Một số thuật toán xấp xỉ cho bài toán tối ưu hàm dạng submodular với ràng buộc***".

2. Ba bài toán quan trọng của tối ưu hàm submodular

Từ các vấn đề nghiên cứu trên đây, có thể thấy rằng có nhiều bài toán tối ưu hàm submodular được quan tâm nghiên cứu hiện nay. Trong đó, có ba bài toán tối ưu hàm submodular có ràng buộc quan trọng và có giá trị ứng dụng trong thực tiễn, còn gọi là các bài toán biến thể của tối đa hàm submodular, được luận án tập trung nghiên cứu:

1. ***Bài toán tối đa hàm k-submodular với ràng buộc chi phí*** (k-Submodular Maximization under Knapsack constraint - kSMK). Sự chuyển hướng đa dạng nghiên cứu với k-submodular được quan tâm vì phù hợp với nhiều lớp bài toán mà hàm submodular chưa đủ để giải quyết, như tập các phần tử được phân loại thành các tập con khác nhau, hoặc được chọn từ nhiều nguồn tài nguyên khác nhau... Các nhà nghiên cứu đã mở rộng hàm submodular thành k-submodular ($k \geq 2$) [125, 110, 109, 126]. Khi đó, hàm mục tiêu sẽ được mở rộng thành $f : (k + 1)^V \mapsto \mathbb{R}_+$. Các ứng dụng của bài toán tối đa hàm k-submodular có thể áp dụng

vào tối đa ảnh hưởng của k chủ đề, tối đa thông tin lan truyền của k chủ đề, đặt k loại cảm biến...

Được mở rộng từ bài toán *tối đa hàm submodular với ràng buộc chi phí (SMK)*, bài toán kSMK xem xét tối đa hàm mục tiêu k-submodular với ràng buộc chi phí cho trước.

2. ***Bài toán tối đa hàm submodular với ràng buộc chi phí có nhiễu*** (Submodular Maximization subject to Knapsack constraint under Noises -SMKN). Bài toán tối đa hàm submodular với ràng buộc chi phí, SMK, được xem là tổng quát hơn so với ràng buộc lực lượng. Ràng buộc này yêu cầu mỗi một phần tử e trong tập dữ liệu đầu vào V sẽ có một chi phí dương $c(e) > 0$ để hoạt động, do vậy, cần phải tìm lời giải sao cho giá trị lợi ích $f(\cdot)$ thu được là lớn nhất mà tổng chi phí của tập lời giải $c(S)$ không vượt quá ngân sách cho trước. Nghiên cứu giải bài toán SMK có khả năng áp dụng vào nhiều trường hợp trong thực tế khi kinh phí, thời gian hay con người bị giới hạn [146, 1, 67, 24].

Tuy nhiên, việc tính toán chính xác hàm mục tiêu f là bất khả thi, nên cần ước lượng xấp xỉ F với sai số $\epsilon \in (0, 1)$ cho nó. F còn gọi là ước lượng nhiễu của f. Tuy nhiên, các nghiên cứu hiện nay chưa xử lý nhiễu khi tính toán với F. Cho nên luận án đặt vấn đề giải bài toán SMK nhưng có xử lý nhiễu qua bài toán SMKN.

3. ***Bài toán Phủ Submodular trên lưới nguyên*** (DR-Submodular Cover over integer lattice - DRSC). Trong khoa học máy tính, bài toán đối ngẫu của tối đa hàm submodular là Phủ Submodular (Submodular Cover - SC) cũng có nhiều ứng dụng có giá trị [145, 98, 107].

Nếu như tối đa hàm submodular tương đương với tìm phủ cực đại, thì SC là bài toán tìm tập phủ tối thiểu sao cho giá trị hàm $f(\cdot)$ không thấp hơn một ngưỡng $\alpha > 0$. Giá trị của bài toán SC thể hiện khi nó tổng quát hóa các trường hợp không đòi hỏi lợi ích thu về cao nhất mà chỉ cần đạt mức nào đó, nhưng phải đảm bảo tối ưu về chi phí con người, ngân sách, thời gian...

Tuy nhiên, SC xét hàm f là hàm tập hợp chưa giải quyết được tình huống *một phần tử tốt có thể được chọn đi chọn lại nhiều lần*. Ví dụ đại lý bán chạy thì nhà phân phối sẽ chọn đi chọn lại để phân phối sản phẩm. Các trường hợp tương tự như vậy cần đưa bài toán lên lưới nguyên (integer lattice) để giải. Hàm submodular mở rộng trên lưới nguyên với $f : 2^V \mapsto \mathbb{R}_+$ trở thành $f : \mathbb{Z}^V \mapsto \mathbb{R}_+$. Submodular đã được mở rộng thành *DR-submodular* và *lattice submodular* trên lưới nguyên với một số khác biệt về tính chất [128]. Bài toán SC lúc này được tổng quát thành *Phủ DR-submodular (DR-Submodular Cover - DRSC)*.

3. Các thách thức đặt ra

Theo xu thế, luận án lựa chọn hướng nghiên cứu các thuật toán xấp xỉ giải bài toán tối đa hàm submodular có ràng buộc. Tuy nhiên, thách thức chung của hướng nghiên cứu này là:

1. Các thuật toán phải phải đối mặt với vấn đề thời gian chạy khi dữ liệu đầu vào tăng lên. Ban đầu, Nemhauser và Wolsey [101, 102] đã chứng minh được giải thuật tham lam giải bài toán tối đa hàm submodular cho lời giải xấp xỉ với tỉ lệ tốt nhất, đạt $(1 - 1/e) \approx 63\%$ lời giải tối ưu. Tuy nhiên, phương pháp tham lam làm bùng nổ số tổ hợp cần tìm nên thuật toán này trở nên bất khả thi với dữ liệu lớn. Chính vì thế, nghiên cứu hiện nay rất cần các thuật toán cải tiến nhằm giảm giải quyết vấn đề thời gian chạy;

2. Các bài toán tối đa hàm submodular thường là các bài toán NP-khó, NP-đầy đủ, thậm chí việc tính toán hàm mục tiêu còn là #P-Khó (bài toán tối đa ảnh hưởng [43]). Các bài toán này đều khó tìm lời giải trong thời gian đa thức. Cộng vào đó là sự khó khăn khi dữ liệu đầu vào ngày càng lớn. Vì thế, người ta thường đưa ra một hàm ước lượng xấp xỉ F của hàm mục tiêu f với sai số $\epsilon \in (0, 1)$. Ước lượng này còn gọi là ước lượng nhiễu. Thời gian giải quyết bài toán bị ảnh hưởng đáng kể bởi nhiễu làm nảy sinh vấn đề cần xử lý nhiễu trong bài toán.

3. Nhiều ứng dụng trong thực tiễn cần mở rộng hoặc biến thể bài toán tối ưu hàm submodular và cần các phương pháp giải phù hợp. Nghiên cứu bài toán tối ưu hàm submodular do vậy cần có sự mở rộng không gian tìm kiếm, xét với nhiều điều kiện ràng buộc khác nhau, xử lý nhiễu, nghiên cứu biến đổi về mặt tính chất của hàm mục tiêu... để đáp ứng ngày càng nhiều hơn các kịch bản khác nhau trong thực tiễn. Do đó, các nghiên cứu giải các bài toán biến thể, bài toán mở rộng của tối ưu hàm submodular là cần thiết.

Đặc biệt, khi lựa chọn ba bài toán nghiên cứu kSMK, SMKN và DRSC, mỗi bài toán lại có những thách thức riêng:

1. ***Đối với bài toán kSMK***.

- Sự khác nhau về bản chất của hàm submodular và hàm k-submodular dẫn tới áp dụng phương pháp đã có với submodular sang k-submodular chưa chắc đã khả thi hoặc hiệu quả bị giảm xuống;

- Ràng buộc chi phí làm cho có nhiều lời giải dự tuyển với nhiều chi phí khác nhau, việc chọn giải pháp tốt nhất giữa rất nhiều giải pháp làm ảnh hưởng đến thời gian chạy của thuật toán;

- Vấn đề giảm độ phức tạp truy vấn, góp phần giải quyết vấn đề thời gian chạy cần nhiều đóng góp mới;

- Vấn đề hàm mục tiêu không đơn điệu trở nên thách thức hơn do phải nhanh chóng loại bỏ các phần tử làm giảm chất lượng hàm mục tiêu.

2. ***Đối với bài toán SMKN***

- Bài toán SMKN là một bài toán mới, chưa có công bố nào về bài toán này;
- Thách thức của ràng buộc chi phí (giống bài toán kSMK) và ước lượng nhiễu F của hàm mục tiêu f;
- Vấn đề thời gian chạy và không gian lưu trữ.

3. ***Đối với bài toán DRSC***

- Hàm submodular trên lưới nguyên có sự biến đổi về tính chất. Đây là một vấn đề còn rất mới, các nghiên cứu về nó chưa nhiều [128, 87, 129];
- DRSC yêu cầu tìm kiếm lời giải trong không gian $\mathbb{Z}$ chiều, lớn hơn rất nhiều so với không gian 2 chiều của hàm tập hợp submodular. Do vậy, việc đề xuất thuật toán xấp xỉ cạnh tranh với độ phức tạp truy vấn tuyến tính rất khó triển khai trên lưới nguyên.

4. Mục tiêu nghiên cứu được đặt ra

Luận án chọn chủ đề nghiên cứu: "***Một số thuật toán xấp xỉ cho bài toán tối ưu hàm dạng submodular với ràng buộc***". Trong đó, do tính độc đáo, giá trị thực tiễn lớn của ba bài toán đã nêu, cùng với các thách thức của chúng, luận án đặt vấn đề nghiên cứu các thuật toán xấp xỉ để giải quyết các bài toán này. Trong đó, mục tiêu chung là các thuật toán xấp xỉ phải hiệu quả với tỉ lệ xấp xỉ cạnh tranh và giảm độ phức tạp truy vấn. Mục tiêu cụ thể như sau:

1. Nghiên cứu bài toán 1 - kSMK với các mục tiêu:

- Đề xuất các thuật toán xấp xỉ giải quyết bài toán kSMK;
- Giải bài toán tổng quát. Các thuật toán đề xuất cho lời giải với tỉ lệ xấp xỉ hằng số cạnh tranh, và giảm độ phức tạp truy vấn xuống còn tuyến tính hoặc giả tuyến tính;
- Thực nghiệm trên một vài kịch bản cho thấy sự tiệm cận giữa lý thuyết với thực nghiệm.

2. Nghiên cứu bài toán 2 - SMKN với các mục tiêu:

- Xây dựng mô hình bài toán tối đa hàm submodular với ràng buộc chi phí dưới sự tác động của 2 loại nhiễu: nhiễu cộng và nhiễu nhân;
- Đề xuất các thuật toán Luồng xấp xỉ hiệu quả giải quyết bài toán SMKN. Thuật toán cải tiến cần cho tỉ lệ xấp xỉ cạnh tranh nhưng giảm thời gian chạy và dung lượng lưu trữ các phần tử so với thuật toán tham lam;
- Ước lượng xấp xỉ hàm mục tiêu trong môi trường nhiễu;
- Thực nghiệm trên một vài kịch bản cho thấy sự tiệm cận giữa lý thuyết với

thực nghiệm.

3. Nghiên cứu bài toán 3 - DRSC với các mục tiêu:

- Nghiên cứu tính chất DR-submodular và mô hình hóa bài toán DRSC;

- Đề xuất thuật toán xấp xỉ hiệu quả giải quyết bài toán DRSC. Với bài toán này, hướng thuật toán đề xuất dựa trên thiết kế song song với độ phức tạp truy vấn và độ phức tạp song song thấp.

5. Phương pháp nghiên cứu

Trên cơ sở của những điều kiện và đặc điểm nghiên cứu có liên quan, cách tiếp cận hợp lý để đạt được mục tiêu nghiên cứu là việc sử dụng các phương pháp nghiên cứu bao gồm:

- Phân tích các nguồn tài liệu như bài báo, sách, bài giảng...;

- Phân tích và tổng hợp lý thuyết về các phương pháp thiết kế thuật toán, đặc biệt là thuật toán xấp xỉ, các phương pháp cải tiến;

- Phương pháp chuyên gia: học hỏi, tham khảo đóng góp, góp ý của các chuyên gia, các nhà khoa học, các giảng viên có kinh nghiệm về giải quyết bài toán tối đa hàm submodular và công tác nghiên cứu;

- Phương pháp thực nghiệm khoa học: xây dựng thực nghiệm trên các ứng dụng cụ thể, kiểm thử chương trình, nhận xét và cải tiến phù hợp.

Từ các phương pháp nêu trên, chúng tôi đã tiến hành khảo sát, phân tích các công trình đã công bố có liên quan. Trên cơ sở đó, luận án đề xuất các thuật toán mới giải quyết các bài toán tối ưu hàm submodular có ràng buộc.

Đặc biệt, các đề xuất mới đều được phân tích đánh giá, chứng minh chặt chẽ thông qua các phân tích lý thuyết được phát biểu dưới dạng các bổ đề, định lý, hệ quả. Ngoài ra, các kết quả lý thuyết còn được kết hợp với các phương pháp thực nghiệm trên máy tính với các bộ dữ liệu chuẩn đã được công bố ở các công trình nổi tiếng, với các thiết lập tham số khác nhau nhằm đảm bảo tính khách quan về hiệu quả của phương pháp đề xuất.

6. Những đóng góp chính và bố cục của luận án

Luận án đã thể hiện những đóng góp sau:

1. Đề xuất các thuật toán xấp xỉ giải quyết bài toán kSMK trong hai trường hợp hàm mục tiêu k-submodular đơn điệu và không đơn điệu.

- Với hàm đơn điệu, luận án đề xuất 03 thuật toán xấp xỉ với độ phức tạp truy vấn tuyến tính. Trong đó, thuật toán tốt nhất nâng được tỉ lệ xấp xỉ so với thuật toán tốt nhất hiện nay mà vẫn giảm độ phức tạp truy vấn xuống một hệ số.

- Với hàm mục tiêu không đơn điệu, luận án đưa ra được một thuật toán cải tiến

cho tỉ lệ xấp xỉ tốt tương đương thuật toán xấp xỉ tốt nhất hiện nay nhưng giảm độ phức tạp truy vấn xuống một hệ số.

- Các thực nghiệm cho thấy các thuật toán của luận án cho chất lượng lời giải cạnh tranh, giảm thời gian chạy đáng kể, đặc biệt khi dữ liệu đầu vào và ngân sách tăng lên.

Các kết quả nghiên cứu đã được công bố tại: hội thảo quốc tế *the 9th IEEE International Conference on Data Science and Advanced Analytics* (DSAA), 2022 (**RANK A**), tạp chí *Computers & Operations Research* (**ISI/Q1**), và hội thảo quốc tế *the 15th IEEE International Conference on Knowledge and Systems Engineering (KSE)*, 2023.

2. Đề xuất các thuật toán xấp xỉ giải quyết bài toán SMKN với hàm mục tiêu submodular đơn điệu. Luận án đề xuất 02 thuật toán, một là thuật toán tham lam và hai là thuật toán luồng cải tiến thuật toán tham lam với các đảm bảo lý thuyết tương đương và giải quyết vấn đề thời gian chạy, không gian lưu trữ. Với nghiên cứu này, luận án cũng trình bày phần thực nghiệm cho thấy thuật toán cải tiến cho giá trị hàm mục tiêu tốt gần bằng tham lam, trong khi thời gian chạy và bộ nhớ tiết kiệm hơn.

Các kết quả nghiên cứu đã được công bố ở tạp chí *Asia-Pacific Journal of Operational Research*, tập 39, số 6, 2022 (**ISI/Q3**).

3. Đề xuất một thuật toán xấp xỉ tiêu chí kép được thiết kế song song hoá giải quyết hai lớp bài toán, DRSC và SC. Thuật toán cho chất lượng lời giải tốt tương đương với thuật toán tiêu chí kép tốt nhất hiện nay cho DRSC, song thuật toán giảm được đáng kể số lượng truy vấn, số lượng vòng tuần tự, qua đó giảm được thời gian chạy.

Các kết quả nghiên cứu đã được công bố tại tạp chí *Information Processing Letters*, tập 182, 2023 (**ISI/Q3**).

Ngoài phần mở đầu và kết luận, luận án chia thành 04 chương:

Chương 1 trình bày bài toán tối ưu hàm dạng submodular, tính chất hàm submodular, các ứng dụng của bài toán tối ưu hàm submodular. Đồng thời, chương này cũng giới thiệu về thuật toán xấp xỉ với các khái niệm về các đảm bảo lý thuyết của thuật toán.

Chương 2 trình bày các kết quả nghiên cứu đối với bài toán tối đa hàm k-submodular với ràng buộc chi phí (Bài toán kSMK). Chương này giới thiệu các thuật toán đề xuất giải quyết bài toán cho hai trường hợp, hàm mục tiêu đơn điệu và không đơn điệu. Ngoài ra, luận án còn trình bày kết quả thực nghiệm trên các bộ dữ liệu thực.

Chương 3 trình bày các kết quả nghiên cứu đối với bài toán tối đa hàm sub-

modular với ràng buộc chi phí trong môi trường có nhiễu (Bài toán SMKN). Hàm mục tiêu f được ước lượng bằng hàm F với hệ số nhiễu $\epsilon \in (0, 1)$ dưới hai mô hình nhiễu cộng và nhiễu nhân. Luận án cũng đề xuất các thuật toán xấp xỉ giải quyết bài toán bằng hai phiên bản, thiết kế tham lam, và phiên bản cải tiến dựa trên thuật toán luồng. Chương này cũng trình bày các kết quả thực nghiệm trên các bộ dữ liệu thực.

Chương 4 trình bày các kết quả nghiên cứu đối với bài toán Phủ Submodular trên lưới nguyên (DRSC). Chương này trình bày thuật toán song song tiêu chí kép dựa trên khái niệm độ phức tạp song song nhằm giảm thời gian chạy. Kết quả về mặt lý thuyết của thuật toán cho thấy tỉ lệ xấp xỉ cạnh tranh, độ phức tạp song song và độ phức tạp truy vấn giảm xuống so với các phương pháp hiện có.

CHƯƠNG 1
BÀI TOÁN TỐI ƯU TỔ HỢP HÀM DẠNG SUBMODULAR

Lớp bài toán quan trọng trong khoa học máy tính là *Tối ưu tổ hợp (Combinatorial Optimization - CO)*, một lớp con của các bài toán *Vận trù học (Operational Research - OR)*. Tối ưu tổ hợp thường được áp dụng làm mô hình để giải cho nhiều bài toán ứng dụng khác nhau như tìm đường đi ngắn nhất, lập lịch, lan truyền tiếp thị [45, 37, 79]... Những bài toán này quan tâm đến tìm một tập lời giải sao cho một yêu cầu nào đó là tốt nhất hoặc tốt hơn một giá trị nào đó.

Tuy nhiên, các bài toán CO hầu hết là NP-khó, NP-đầy đủ nên các kỹ thuật giải chính xác khó tìm được lời giải trong thời gian đa thức, cần phải tìm lời giải gần đúng [79]. Các phương pháp giải gần đúng các bài toán CO gồm có các thuật toán dựa trên kinh nghiệm (heuristic và metaheuristic) và các thuật toán xấp xỉ.

Tuy nhiên, nhược điểm của các phương pháp tìm kiếm dựa trên kinh nghiệm là không chỉ ra đảm bảo lý thuyết so với lời giải tốt nhất thì lời giải này đúng được bao nhiêu %. Đôi khi, thiết kế một thuật toán có thể dẫn tới kết quả tồi, ảnh hưởng tới thời gian kiểm nghiệm, tinh chỉnh các tham số của thuật toán.

Thuật toán xấp xỉ (approximation algorithm) với các đảm bảo lý thuyết được chỉ rõ khắc phục được các nhược điểm nói trên. Giải xấp xỉ tức là đưa ra lời giải gần tối ưu.

Vì vậy, luận án nghiên cứu giải bài toán CO hàm dạng submodular bằng các thuật toán xấp xỉ. Hàm submodular mới được nghiên cứu trong vòng 20 năm và trở nên thu hút trong thời gian gần đây [86, 85, 34, 1, 67, 24, 41, 35, 44]. Sở dĩ bài toán này được quan tâm do tính chất độc đáo của hàm mục tiêu submodular làm cho nhiều bài toán ứng dụng thực tiễn như tóm tắt văn bản [98], đặt cảm biến giám sát chất lượng nước hoặc dự báo thời tiết [80], tối đa ảnh hưởng trên mạng xã hội [43]... có thể giải được dưới góc nhìn của một bài toán tối ưu. Trong đó, bài toán *tối đa hàm submodular, SM*, được đánh giá là một trong các bài toán đặc biệt nhất khi có thể nhanh chóng thu thập và xử lý thông tin mà không tạo ra sự dư thừa, lãng phí hay làm mất mát thông tin [13].

1.1. Bài toán CO tối đa hàm submodular

1.1.1. Phát biểu bài toán

Bài toán CO ứng với một bộ ba $(S, f, \mathcal{C})$ yêu cầu:

$$(\min) \max f(s)$$
$$\text{s.t } s \in S.$$

Hoặc tìm $s^* = \arg(\min) \max_{s \in S} f(s)$. Có nghĩa là tìm tập s sao cho $f(s)$ đạt giá trị lớn nhất hoặc nhỏ nhất với một số ràng buộc $\mathcal{C}$ đối với s. $f(\cdot)$ được gọi là hàm mục tiêu, $s \in S$ là lời giải hoặc phương án chấp nhận được, và S là tập các phương án được xét. s^* được gọi là lời giải tối ưu. Với bài toán tìm giá trị lớn nhất thì $f(s^*) \geq f(s)$ với mọi lời giải s. Bài toán SM phát biểu dưới dạng một bài toán CO như sau:

Định nghĩa 1.1 (Bài toán SM)**.** Cho tập cơ sở V và hàm $f : 2^V \mapsto \mathbb{R}_+$ là hàm tập hợp submodular (với $f(\emptyset) = 0$), bài toán cần tìm:

$$\begin{aligned} &\max f(S) \\ &\text{s.t } S \subseteq V, \end{aligned} \tag{1.1}$$

với $\mathcal{C}$ là ràng buộc cho trước của bài toán.

Một số trường hợp phổ biến của $\mathcal{C}$ là:

- $\mathcal{C} = \emptyset$: bài toán không ràng buộc;
- $\mathcal{C} = \{|S| \leq k\}$: ràng buộc lực lượng;
- $\mathcal{C} = \{\forall A \subseteq B \subseteq V, f(A) \leq f(B)\}$: polymatroid (hàm f có tính đơn điệu)};
- $\mathcal{C} = \{c(S) \leq B\}$ với $c(S)$ là chi phí của tập S, B là ngân sách cho trước, đây là ràng buộc chi phí.

Một lưu ý khi giải các bài toán SM, đó là người ta luôn giả sử đã cho trước cách tính hàm f. Nói cách khác, tồn tại một hộp đen sao cho với mọi đầu vào $S \subseteq V$ qua hộp đen này đều tính được $f(S)$. Giá trị $f(S)$ lúc này được gọi là giá trị *ước lượng (oracle)* của hàm mục tiêu, hoặc nói ngắn gọn là giá trị của hàm mục tiêu. Hiện nay, người ta rất quan tâm đến số lời gọi các ước lượng này và đưa ra một phép đo quan trọng để đánh giá độ phức tạp của thuật toán là *số lượng truy vấn (query)* được thể hiện bằng *độ phức tạp truy vấn (query complexity)*.

Điều làm nên sự đặc biệt của bài toán SM là tính chất submodular của hàm mục tiêu. Phần tiếp theo sẽ nghiên cứu sâu hơn về hàm này.

1.1.2. Hàm mục tiêu submodular

1.1.2.1. Định nghĩa

Phần lớn các bài toán tối ưu hàm dạng submodular làm việc với hàm submodular là hàm tập hợp [90, 94, 75, 111, 10, 3, 60, 52, 19]... Việc ứng dụng hàm submodular cho nhiều lớp bài toán cho thấy hàm này có tính chất rất mạnh mẽ và cho nhiều lợi ích khi ứng dụng vào trong tối ưu.

Về mặt lý thuyết, hàm submodular được phát triển vào những năm đầu của

những năm 50 bởi các nhà khoa học như H.Whitney và W.T. Tutte, G.Choquet và O.Ore ... Đến những năm 60, lý thuyết về submodular bắt đầu nở rộ với các nghiên cứu của J.Edmonds xét với ràng buộc matroid và polymatroid [58]. Fujishige đã giới thiệu một cách chi tiết lịch sử hình thành, phát triển các nghiên cứu về hàm này [57, 58]. Hàm submodular là một hàm trên một lưới phân tán, một mạng con của một lưới Boolean [1].

Hàm *submodular*, còn gọi là hàm có tính chất *submodularity*, được phát biểu như sau:

Định nghĩa 1.2 (Hàm submodular [59])**.** Cho V là một tập hữu hạn không rỗng và $\mathcal{D}$ là một tập hợp các tập con của V tạo thành một lưới phân phối với các phép toán $\cup, \cap$ là các toán tử trên lưới. Có nghĩa là với mỗi tập $X, Y \in \mathcal{D}$ chúng ta có $X \cup Y, X \cap Y \in \mathcal{D}$. Đặt $f : \mathcal{D} \mapsto \mathbb{R}$ là một hàm submodular trên lưới phân phối $\mathcal{D}$, thì f thỏa mãn bất đẳng thức sau:

$$\forall X, Y \in \mathcal{D} : f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y). \tag{1.2}$$

Hàm submodular được xây dựng trên một họ $\mathcal{D}$ các tập con của V, khi đó, bất đẳng thức (1.2) yêu cầu $X, Y, X \cap Y, X \cup Y \subseteq \mathcal{D}$. Vì $\mathcal{D} \subseteq 2^V$ nên hàm mục tiêu f trở thành: $f : 2^V \mapsto \mathbb{R}$ với V còn gọi là tập cơ sở. Do vậy, bài toán SM có hàm mục tiêu submodular được mô tả như Định nghĩa 1.3 dưới đây.

Định nghĩa 1.3 (Hàm submodular)**.** Cho V là một tập hữu hạn không rỗng được gọi là tập cơ sở, hàm $f : 2^V \mapsto \mathbb{R}_+$ là một hàm submodular khi và chỉ khi nó thỏa mãn bất đẳng thức sau:

$$\forall X, Y \subseteq V : f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y). \tag{1.3}$$

Không giảm tổng quát, hàm $f : 2^V \mapsto \mathbb{R}_+$, tức là $f(S) \geq 0, \forall S \subseteq V$. Lý giải là vì với f bất kỳ ta hoàn toàn có thể cộng thêm cho nó một hằng số c để $f(\cdot) \geq 0$ mà tính chất submodular vẫn được bảo toàn [57]. Thêm vào đó, luôn có giả sử $f(\emptyset) = 0$ biểu thị đối với tập rỗng, hàm không thu được một lợi ích nào.

Đối ngẫu với submodular là hàm supermodular [2]. Trong luận án, một số thuật ngữ viết bằng tiếng Anh như *modular, submodular, submodularity, matroid, polymatroid*... sẽ được giữ nguyên.

[1] Một lưới Boolean $\mathbf{B}$ là một lưới phân phối trong đó mỗi phần tử $x \in \mathbf{B}$ đều có một phần bù $\overline{x} \in \mathbf{B}$ sao cho $x \wedge \overline{x} = 0, x \vee \overline{x} = 1, \overline{\overline{x}} = x, (x \wedge y) = \overline{x} \vee \overline{y}, \overline{x \vee y} = \overline{x} \wedge \overline{y}$

[2] Hàm có tính chất supermodular là hàm có tính chất: $\forall X, Y \in \mathcal{D} : g(X)+g(Y) \leq g(X \cap Y)+g(X \cup Y)$. Khi dấu $=$ của bất đẳng thức xảy ra, ta nói hàm có tính modular.

1.1.2.2. Tính chất lợi nhuận hiệu suất giảm dần

Khi nghiên cứu, hàm submodular được chỉ ra có sự phù hợp với *tính chất lợi nhuận hiệu suất giảm dần*, một tính chất rất nổi tiếng thường dùng trong kinh tế học. Chẳng hạn đối với doanh thu của một cửa hàng, giá trị đóng góp về mặt lợi nhuận của một mặt hàng bán được đối với tổng doanh thu của cửa hàng khi bán được ít hàng sẽ lớn hơn khi cửa hàng đã bán được nhiều hàng hơn. Điều này cũng giống như khi ai đó chưa có đồng tiền nào trong tài khoản, thu nhập thêm 1 triệu đồng có ý nghĩa hơn, hay có giá trị đóng góp lớn hơn khi tài khoản đã có hàng trăm triệu đồng. Tối đa doanh thu hay tối đa lợi ích là các thể hiện rất cụ thể của tối đa hàm submodular (Hình 1.1).

Giả sử tập $A \subseteq V$ là một tập các người dùng hoặc hoạt động đầu tư nào đó cho lợi ích là $f(A)$. Hàm submodular f mang ý nghĩa sau khi thực hiện một loạt các hành động của tập A, *lợi nhuận biên (marginal gain)* [3] của bất kì một phần tử e nào đó sẽ không tăng khi thực hiện các hoạt động trong tập $V \setminus A$.

Do đó, hàm submodular trên tập hợp là hàm có tính chất *lợi nhuận hiệu suất giảm dần (diminishing returns property)* tự nhiên được phát biểu như sau:

Định nghĩa 1.4. (Tính chất lợi nhuận hiệu suất giảm dần của hàm submodular [13]) Hàm tập hợp $f : 2^V \mapsto \mathbb{R}_+$ là hàm submodular và các tập $A \subseteq B \subseteq V$, với mọi $e \notin B$ ta có lợi nhuận biên của e khi đóng góp vào tập nhỏ hơn, A, sẽ ít nhất bằng lợi nhuận biên khi đóng góp e vào tập lớn hơn, B:

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B). \tag{1.4}$$

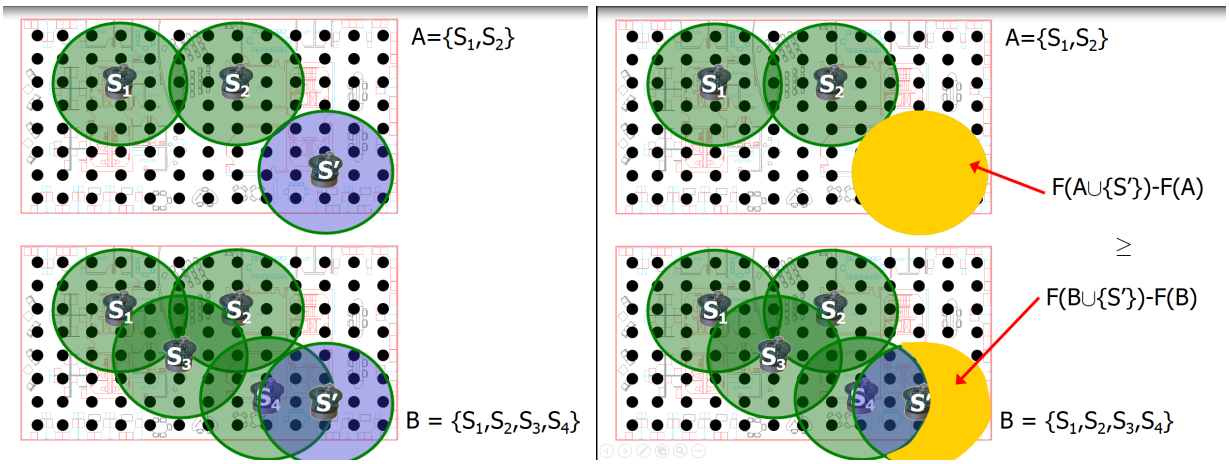
Hình 1.1 minh họa cho hàm mục tiêu có tính chất lợi nhuận hiệu suất giảm dần trong bài toán đặt cảm biến. Đóng góp lợi ích có được bởi đặt một cảm biến S' trên một sơ đồ đã đặt các cảm biến S_1, S_2 không tăng khi chúng ta đặt tại sơ đồ có thêm các cảm biến S_3, S_4.

1.1.2.3. Hàm submodular đơn điệu

Một lớp bài toán con quan trọng khi nghiên cứu về hàm submodular đó là hàm có *tính chất đơn điệu tăng*, nói ngắn gọn là tính chất đơn điệu (monotone). Hàm đơn điệu nói rằng khi tăng kích cỡ của tập đối số thì giá trị của hàm không bị giảm đi.

Định nghĩa 1.5. (Tính chất đơn điệu [80]) Hàm $f : 2^V \mapsto \mathbb{R}_+$ là hàm submodular

[3] Đôi khi các tài liệu còn sử dụng các thuật ngữ tương tự như marginal profit, marginal benefit, hoặc contribution gain.



Hình 1.1: Bài toán tìm vị trí đặt cảm biến hàm mục tiêu có tính chất lợi nhuận hiệu suất giảm dần [81].

đơn điệu nếu với mọi tập $A \subseteq B \subseteq V$ ta có $f(A) \leq f(B)$.

Lưu ý là hàm f sẽ chỉ đơn điệu khi và chỉ khi tất cả các lợi nhuận biên của mọi tập con đều không âm, tức là với mọi $\forall A \subseteq B \subseteq V$ và $e \in V$ thì có:

$$\Delta(e|A) \geq \Delta(e|B) \tag{1.5}$$

Bất đẳng thức này hơi khác một chút với bất đẳng thức ở Định nghĩa 1.4. Vì định nghĩa phải đưa thêm ràng buộc $e \notin B$, với hàm submodular đơn điệu thì ràng buộc này không cần nữa.

Một hàm là submodular đơn điệu tăng, chuẩn hóa $f(\emptyset) = 0$, thì hàm đó là hàm polymatroid [59]. Hàm Entropy được xem là một hàm polymatroid nhưng nó không bao gồm mọi hàm polymatroid. Hàm Entropy được sử dụng để đo chất lượng thông tin do cảm biến thu được trong bài toán đặt cảm biến [81].

Ngoài tính chất lợi nhuận hiệu suất giảm dần, tính đơn điệu, hàm mục tiêu submodular còn có thể biểu hiện thành nhiều tính chất khác như tính phân rã, tính cắt bớt, tính thặng dư [80, 13]... Sự phong phú, linh hoạt của hàm submodular khiến cho nó có thể áp dụng vào giải nhiều bài toán khác nhau với các phương pháp giải khác nhau được vận dụng trong thực tiễn và trong nghiên cứu.

1.2. Lợi ích và ứng dụng của tối đa hàm submodular

1.2.1. Lợi ích của tối đa hàm submodular

Bài toán tối đa hàm submodular, SM, thường được áp dụng khi cần thu thập thông tin nhiều nhất có thể, hay khi ai đó muốn khuyến khích tính đa dạng, tính độc lập, có sự lan truyền hoặc phân tán thông tin [13]. Sự linh hoạt của hàm

submodular làm cho thông tin có thể được thu thập và xử lý hiệu quả. Cụ thể, bài toán SM mang lại các lợi ích trong ứng dụng như sau:

Đầu tiên, *hàm submodular có thể hoạt động giống như một dạng hàm thông tin*. Chẳng hạn hàm thông tin phụ thuộc lẫn nhau là một hàm có tính submodular [80]. Vì thế, bài toán SM có thể mô hình hóa cho các bài toán cần thu thập thông tin hỗ trợ công tác dự báo và ra quyết định như đặt cảm biến [81], xây dựng các hệ thống gợi ý [107, 111, 64], lan truyền thông tin [43, 105, 25]...

Thứ hai, *cực đại hàm submodular tạo ra sự đa dạng của thông tin*. Xu hướng của hàm submodular khi chọn một phần tử mới thêm vào tập lời giải là nó sẽ chọn phần tử nào khác biệt nhất so với các phần tử đã chọn. Do vậy, *cực đại một hàm submodular sẽ chọn ra các phần tử khác nhau nhất*. Các phần tử giống với các phần tử đã được chọn sẽ mất đi giá trị so với giá trị cốt lõi của nó, trong khi các phần tử khác chưa chọn sẽ không bị giảm đi.

Hệ quả là *bài toán SM sẽ chọn ra tập con không dư thừa và vì vậy không lãng phí*. Việc cực đại một hàm submodular thành công liên quan đến lựa chọn các phần tử khác nhau, đó chính là định nghĩa của sự đa dạng. Sự đa dạng nói chung là một khía cạnh cực kì quan trọng trong học máy và trí tuệ nhân tạo. Sự thiên vị trong khoa học dữ liệu và học máy thường có thể được coi là sự thiếu đa dạng ở đâu đó. Hàm submodular có khả năng khuyến khích (và thậm chí đảm bảo) sự đa dạng, tăng cường sự cân bằng và giảm sự sai lệch trong trí tuệ nhân tạo. Tuy nhiên, để một hàm submodular có thể mô hình sự đa dạng một cách thích hợp, nó cần phải được khởi tạo một cách phù hợp.

Thứ ba, *bài toán SM sẽ hạn chế sự mất mát thông tin*. Nếu tìm ra một tập nhỏ mà có thể đại diện cho gần hết hoặc tất cả các giá trị của toàn thể (chính là cực đại hóa) thì thông tin sẽ không bị mất mát khi đo bằng hàm submodular. Khi đó tập con được chọn tuy cỡ nhỏ mà hiệu quả khi biểu diễn toàn bộ thông tin của cả tập dữ liệu có kích cỡ lớn hơn rất nhiều. Việc áp dụng SM vào các bài toán chọn tập con đặc trưng đã được khẳng định phát huy hiệu quả [120].

Từ đó, cho thấy một ưu điểm quan trọng của SM, là *giúp giảm được kích thước dữ liệu*. Nhiều ứng dụng như tự động tóm tắt tài liệu [99, 94, 5], tóm tắt hình ảnh [86, 34]... đã được chỉ ra có thể áp dụng mô hình bài toán SM để giải. Điều này là hết sức cần thiết với kỷ nguyên của dữ liệu lớn và nhu cầu thu thập và phân tích thông tin ngày càng cấp thiết hiện nay.

Dữ liệu thu thập từ nhiều nguồn khác nhau ngày càng khổng lồ là nguồn đầu vào quý giá để đưa vào các mô hình huấn luyện phục vụ các bài toán như nhận dạng, xử lý ngôn ngữ tự nhiên... Tuy nhiên, nó cũng đòi hỏi một chi phí lớn để thu thập, lưu trữ, phân tích, gán nhãn cho các dữ liệu. Hơn nữa, dữ liệu thu thập

càng nhiều, khả năng dữ liệu bị trùng lặp lại càng lớn gây ra sự lãng phí về mặt tài nguyên, công sức, tiền của...

Bài toán SM chọn ra một tập con S đặc trưng cho cả tập dữ liệu V khổng lồ, với $|S| \ll |V|$ đã giúp cho lượng dữ liệu cần xử lý được giảm đi đáng kể. Nếu V là một tập huấn luyện đủ lớn để huấn luyện cho một mạng nơ-ron học sâu, lời giải của bài toán SM sẽ trả về một tập con S kích thước k nhỏ hơn nhiều kích thước của tập V để huấn luyện thay vì cả tập V. Rõ ràng, tập này hoạt động tốt hơn nhiều so với một tập hợp con ngẫu nhiên đồng nhất có kích thước k.

Cuối cùng, việc xử lý dữ liệu dưới góc nhìn của tối ưu hóa, xây dựng bài toán ứng dụng bằng cách đưa về bài toán SM, tìm ra các thuật toán xấp xỉ cải tiến giảm độ phức tạp thời gian, độ phức tạp bộ nhớ *giúp cho ứng dụng được chạy nhanh hơn, giảm bớt yêu cầu về tài nguyên máy tính*. Điều này góp phần tạo ra một hướng đi trong thực nghiệm, khi mà các mô hình huấn luyện thường gặp trong học máy có thể tốn rất nhiều thời gian và tài nguyên để thực hiện.

Từ các lợi ích nói trên, cho thấy ứng dụng bài toán SM có ý nghĩa thiết thực. Phần tiếp theo sẽ giới thiệu một số ứng dụng cụ thể của bài toán này.

1.2.2. Ứng dụng của bài toán SM

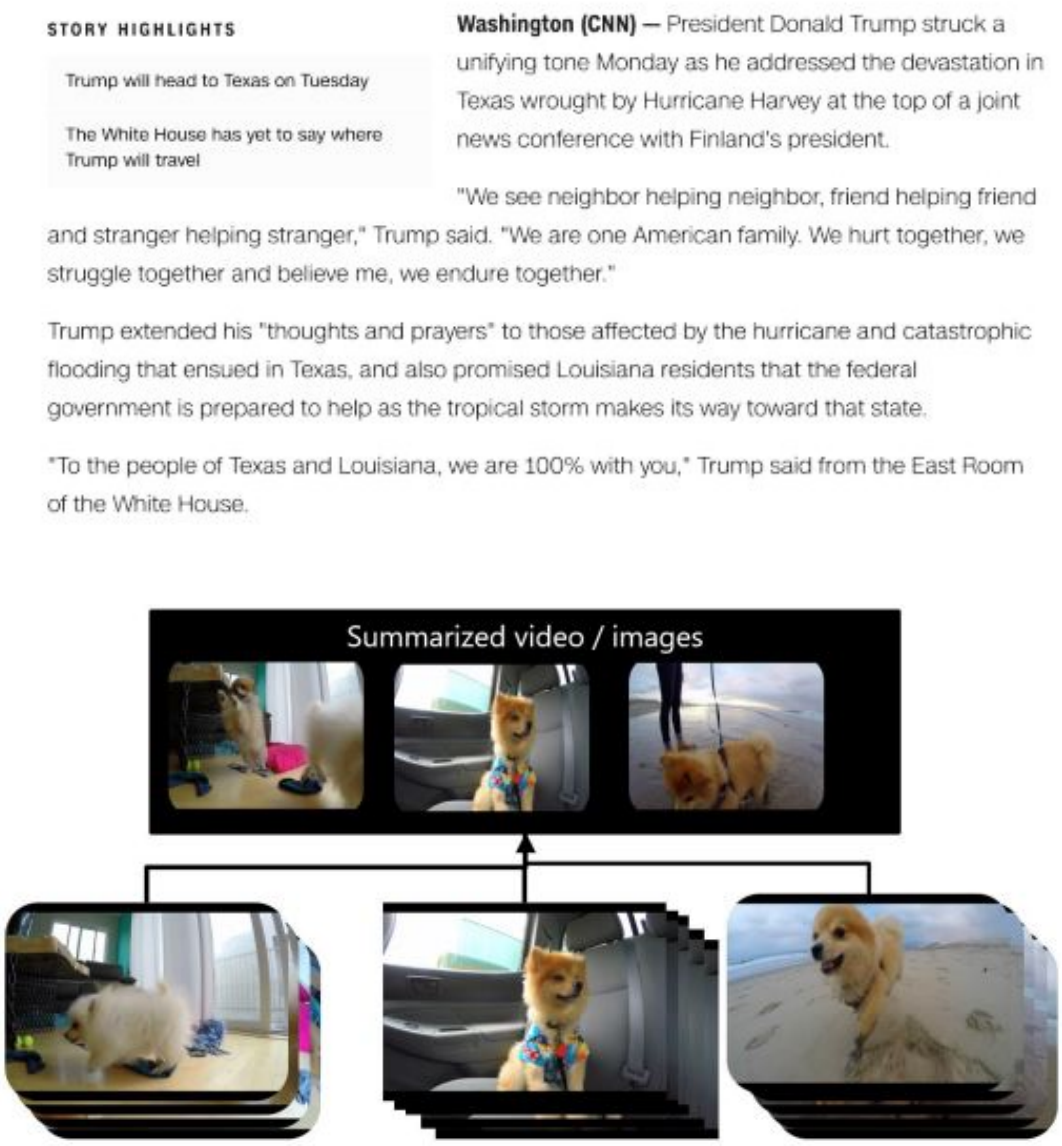
1.2.2.1. Tóm tắt dữ liệu

Tóm tắt dữ liệu có nhiều ứng dụng khác nhau như tóm tắt văn bản, tóm tắt hình ảnh (Hình 1.2)... Đây là các bài toán phổ biến trong học máy hiện nay, xuất phát từ yêu cầu cá nhân hóa, tự động hóa trong dịch vụ và sản xuất trước sự bùng nổ của dữ liệu trực quan. Ví dụ: Các hệ thống gợi ý tìm kiếm, gợi ý phim ảnh theo thị hiếu người dùng.

Với thời đại hiện nay, dữ liệu ngày càng bùng nổ. Do đó các bài toán thu thập và phân tích dữ liệu phải tăng chi phí để xử lý dữ liệu. Từ đó dẫn tới yêu cầu chọn lọc để dữ liệu được dùng hiệu quả và giảm số lượng trùng lặp, dư thừa. Từ đó hình thành lớp bài toán tóm tắt dữ liệu, trích chọn tập con đặc trưng S đại diện cho cả tập dữ liệu V.

Iyer và cộng sự [120] đã trình bày một vài cách tiếp cận cho bài toán này như xấp xỉ gradient, xây dựng bộ huấn luyện, dựa trên phân phối... Đặc biệt, các tác giả cũng chỉ ra rằng các hướng nghiên cứu trên đều có thể đưa về sử dụng *hàm submodular* như một hàm đại diện để lựa chọn tập hợp con dữ liệu. Bài toán tóm tắt dữ liệu khi đó được phát biểu như sau:

Định nghĩa 1.6 (Tóm tắt dữ liệu)**.** Cho một tập cơ sở $V = \{1, 2, \ldots, n\}$. Ta định nghĩa hàm: $f : 2^V \mapsto \mathbb{R}_+$ đo mức đại diện (mức phủ) của một tập con $S \subseteq V$ đối

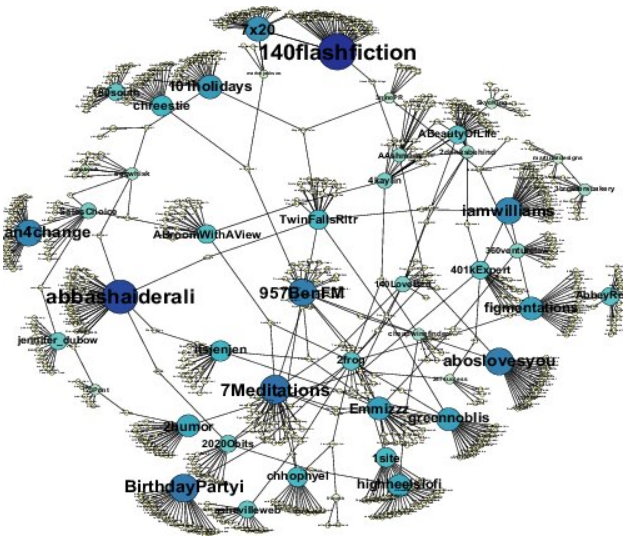


(a) (b)

Hình 1.2: (a) Ứng dụng tóm tắt văn bản trong điểm tin; (b) Tóm tắt hình ảnh nhằm thu nhỏ tập dữ liệu [120].

với tập V. Bài toán yêu cầu cần tìm tập S không quá k phần tử $|S| \leq k$ sao cho phủ (đại diện) của tập S, $f(S)$, đối với tập V là lớn nhất.

Các công bố gần đây trong học máy [5, 98, 99]... đã chứng minh SM là mô hình tốt cho bài toán tóm tắt dữ liệu. Tóm tắt dữ liệu cũng thường xuyên được đưa vào các thực nghiệm nghiên cứu về tối ưu hàm submodular [85, 34, 24].



Hình 1.3: Lan truyền thông tin trên mạng Twitter [142]

1.2.2.2. Tối đa ảnh hưởng trên mạng xã hội

Từ khi mạng xã hội (MXH) xuất hiện thì nhu cầu lan truyền thông tin nhanh chóng trên nó xuất hiện. Do đó, nảy sinh yêu cầu sử dụng MXH để quảng cáo, tiếp thị sản phẩm (tiếp thị lan truyền) cho các tổ chức, doanh nghiệp, hoặc tuyên truyền, vận động tranh cử... Từ đó nảy sinh lớp bài toán lan truyền thông tin hay lan truyền ảnh hưởng (Hình 1.3). Bài toán *tối đa ảnh hưởng (Influence Maximization - IM)* là chủ đề nóng trong khoảng chục năm gần đây [16, 105, 71, 143].

Kempe và cộng sự [43] lần đầu tiên phát biểu bài toán này trên hai mô hình phát tán thông tin là *Bậc độc lập (Cascade Independent - IC)* và *Ngưỡng tuyến tính (Linear Threshold - LT)*, sau này sẽ gọi tắt là 2 mô hình IC và LT. Sau đó, bài toán được nghiên cứu rộng rãi và mở rộng [16, 33, 91, 132, 105, 143, 106]... với các đề xuất nhằm tăng tốc độ thuật toán, thay đổi chiến lược lấy mẫu nhằm giải được với tập dữ liệu có thể lên tới hàng triệu đỉnh và hàng tỉ cạnh. Bài toán được phát biểu như sau:

Định nghĩa 1.7 (Bài toán IM [43])**.** Một MXH được biểu diễn bằng một đồ thị $G = (V, E)$ trên mô hình phát tán thông tin $\mathcal{M}$, trong đó V gồm n đỉnh biểu diễn tập người dùng trên MXH, tập E gồm m cạnh $e = (u, v)$ biểu diễn liên kết giữa 2 người dùng u và v. Cho trước một số nguyên dương $k > 0$, bài toán yêu cầu tìm $S \subseteq V, |S| \leq k$ sao cho ảnh hưởng của tập S trên tập V, $\sigma(S)$, là lớn nhất.

Để giải quyết bài toán IM, Kempe [43] đầu tiên đã chỉ ra được hàm mục tiêu có tính chất *submodular*. Ông sử dụng phương pháp giải xấp xỉ cho tỉ lệ là $1 - 1/e$ bằng thuật toán tham lam Leo đồi kết hợp phương pháp lấy mẫu Monte-Carlo. Sau

đó, Borg và các cộng sự [16] đã tạo ra một bước đột phá trong phương pháp lấy mẫu giúp giảm được thời gian chạy của thuật toán. Ông đã đề xuất một phương pháp gọi là *Lấy mẫu ảnh hưởng ngược (Reverse Influence Sampling - RIS)* xây dựng một mô hình mới biến đổi từ mô hình IC. Kế thừa kết quả của nghiên cứu này, nhiều tác giả đã cải tiến mô hình RIS [133, 132, 105, 71] để giảm số lượng mẫu cần lấy, giảm số vòng lặp, tăng tốc độ thuật toán và giải quyết bài toán IM với dữ liệu đầu vào có thể lên tới hàng trăm triệu đỉnh.

Sau này, Wang và cộng sự [143] đã mở rộng phạm vi ảnh hưởng của các nút bằng khái niệm nút *được thông tin (informed node)*. Các tác giả cho rằng một nút chưa được kích hoạt nhưng nếu nó có ít nhất một hàng xóm là nút đã được kích hoạt thì nó trở thành nút có được thông tin. Như vậy một bài toán biến thể mới của bài toán IM được đề xuất, đó là bài toán *tối đa độ phủ của thông tin (Information Coverage Maxization - ICM*) cần phải cực đại quá trình lan truyền theo kỳ vọng gồm cả những nút được kích hoạt và nút người được thông tin về sự kiện. Các tác giả đã chỉ ra tài toán ICM cũng hoạt động trên mô hình lan truyền thông tin giống IM và hàm mục tiêu cũng có tính chất *submodular*.

Các thách thức của bài toán lan truyền thông tin đó là các bài toán là NP-khó [43], thậm chí tính toán chính xác hàm mục tiêu thuộc lớp bài toán #P-Khó [31, 33]. Tuy nhiên, do tính ứng dụng cao của IM trong kinh tế, xã hội nên nó thu hút được rất nhiều sự quan tâm của giới nghiên cứu trong thời gian gần đây. Các công bố có thể là phát triển thuật toán [42, 16, 92, 105, 112] hoặc nghiên cứu các bài toán biến thể của nó [17, 29, 12, 103, 150, 96]...

1.2.2.3. Tối đa hoá doanh thu

Đây là bài toán biến thể từ bài toán IM ở trên. Hartline và cộng sự [68] lần đầu tiên đặt vấn đề xây dựng mô hình tối đa hóa doanh thu từ người mua hàng, qua đó thực hiện một chiến dịch chào hàng trên MXH. Bài toán này xem xét việc một người bán hàng muốn bán một loại hàng hóa đến một nhóm người dùng tiềm năng V. Người bán cần tối đa hóa doanh thu của mình. Giả sử rằng người bán không biết đến mức chi trả tối đa của người mua mà chỉ biết mức phân phối F từ các lợi ích thu về. F là một phân phối tích lũy giá trị của người mua, tức là $F(t)$ là xác suất lợi ích từ người mua nhỏ hơn giá trị t. Từ đó, định nghĩa về bài toán *tối đa doanh thu (Revenue Maximization - RM)* được phát biểu như sau:

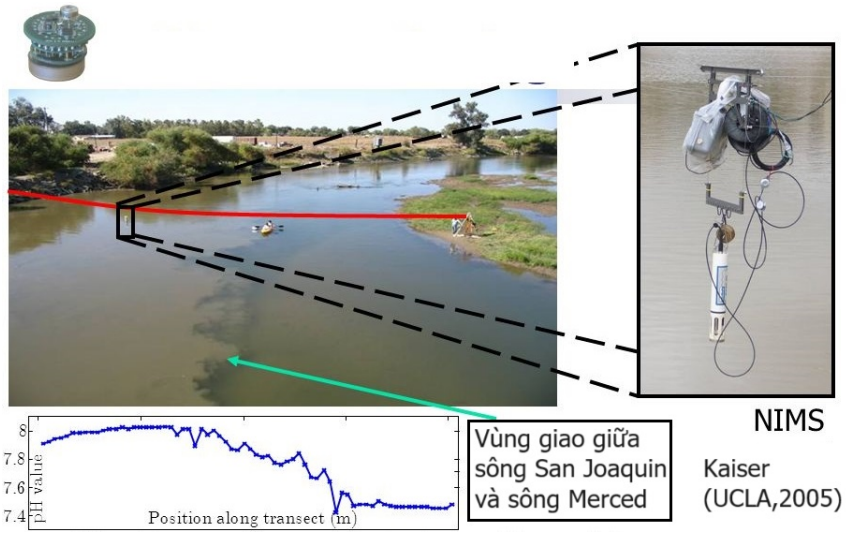
Định nghĩa 1.8 (Bài toán RM [68])**.** Giả sử rằng lợi ích của người mua được phân phối theo phân phối F. Giá tối ưu p^* cực đại doanh thu theo kỳ vọng có được từ người mua i nào đó, có nghĩa là giá p^* cực đại hàm $p.(1 - F(p))$. Doanh thu tối ưu là $p^*.(1 - F(p^*))$ theo kỳ vọng.

Khi áp dụng thuật toán để giải cho bài toán RM, các tác giả cũng đã chỉ ra mô hình đồ thị lõm là phù hợp, đưa bài toán về tối đa *hàm submodular đơn điệu* và sử dụng điều kiện tỷ lệ rủi ro đơn điệu. Các kết quả nghiên cứu của họ sau này được áp dụng trong nhiều thực nghiệm đối với bài toán tối đa hàm submodular [85, 34, 24].

1.2.2.4. Đặt cảm biến tối đa thông tin thu được

Bài toán *đặt cảm biến (Sensor Placement - SP)* nhằm tối đa thông tin thu được là một dạng của bài toán thu thập thông tin. Bài toán này xuất phát từ mong muốn tìm hiểu về các vấn của thế giới tự nhiễn như đo chất lượng nước trong một vùng địa lý (Hình 1.4), phát hiện động đất, phát hiện ô nhiễm môi trường... Chỉ dựa vào các hiện tượng quan sát được: dùng các phép đo, đặt các cảm biến, chọn các thông số thực nghiệm... để thu thập thông tin, từ đó hỗ trợ ra quyết định dự báo cho những điều không thể quan sát được như dự báo thời tiết, cảnh báo động đất...

Chi phí cho các bài toán nghiên cứu trên là rất đắt đỏ, tốn công sức, thời gian... Vì vậy người ta muốn xây dựng phương pháp học sao cho có thể tối ưu được chi phí nhưng đạt được lượng thông tin hữu ích nhất. Chẳng hạn như nước bị ô nhiễm ảnh hưởng nghiêm trọng đến môi trường và sự tồn vong của nhân loại. Do vậy, cần phải đặt các cảm biến để phát hiện các loại ô nhiễm (ví dụ asen, vi khuẩn giết người...). Một câu hỏi đặt ra: "Làm sao để nhanh chóng phát hiện ra loại ô nhiễm đó?". Như vậy phải đặt các cảm biến sao cho nó bao quát được toàn bộ vùng cần giám sát (Hình 1.5).



Hình 1.4: Hệ thống cảm biến thông minh NIMS do Kaiser[74] đề xuất

Bài toán đặt cảm biến được nghiên cứu nhiều trong học máy [65, 81, 82, 84, 83, 46]... Bài toán này được Krause và cộng sự mô tả như sau [84]: Cảm biến có vùng cảm biến xác định, như một chiếc đĩa có bán kính nhất định và chỉ có thể

giao tiếp với các cảm biến khác cách nhau tối đa một khoảng xác định nào đó. Khái niệm về một vùng cảm biến giả thiết rằng các cảm biến có thể quan sát mọi thứ một cách hoàn hảo trong khu vực, nhưng không thu được gì ở bên ngoài vùng của nó. Thứ hai, giả định rằng hai cảm biến tại các vị trí cố định có thể giao tiếp một cách hoàn hảo (nghĩa là họ "được kết nối") hoặc hoàn toàn không giao tiếp (bị "ngắt kết nối").

Bài toán tìm vị trí để *đặt cảm biến, SP,* được phát biểu như sau:

Định nghĩa 1.9 (Bài toán SP [81])**.** Cho trước một tập V các vị trí cần giám sát, một cảm biến có khả năng đo một vùng trong vòng tròn bán kính xung quanh nó. Với mỗi tập $A \subseteq V$, hàm mục tiêu $f(A)$ ="Vùng được giám sát bởi các cảm biến đặt tại A". Giả sử tập $\mathcal{U}$ là tập hợp các tập con $S_i \subseteq \mathcal{U}$. Với $A \subseteq V = \{1, 2, .., n\}$, định nghĩa $f(A) = |\bigcup_{i \in A} S_i|$. Bài toán yêu cầu tìm A để vùng A có thể giám sát được, $f(A)$, là lớn nhất.

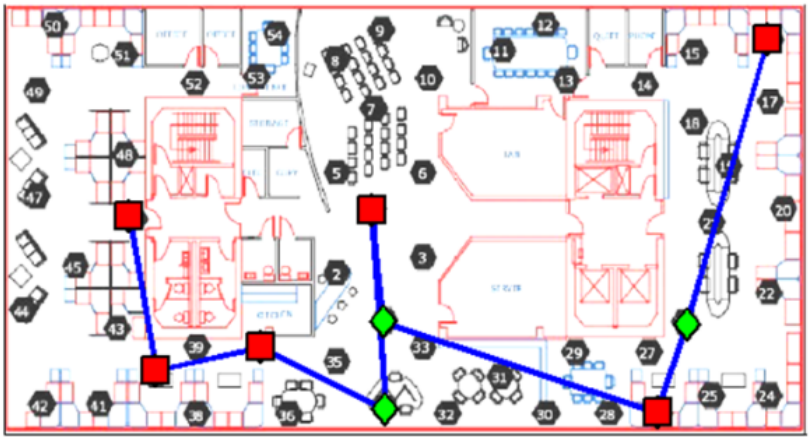
Với định nghĩa được mô tả như trên, bài toán trở thành bài toán tập phủ. Bài toán tập phủ đã được chỉ ra có tính chất *submodular* [145] (Hình 1.1). Hàm $f(A)$ còn gọi là hàm thông tin thu được, hay là hàm đo chất lượng giám sát. Krause và cộng sự [84, 82, 83] cũng đưa ra một phương pháp tính chất lượng giám sát khi xây dựng $f(\cdot)$ là một *hàm đo thông tin phụ thuộc lẫn nhau (mutual information funcion)* giữa các vị trí được chọn và các vị trí chưa được chọn.

Krause đã chỉ ra rằng bằng phép đo như trên, hàm mục tiêu có tính *submodular* đơn điệu không giảm (Hình 1.1) và bài toán tối đa hàm mục tiêu là NP-khó. Các tác giả cũng đã đưa ra một phép tính xấp xỉ trong thời gian đa thức cho lời giải xấp xỉ $1 - 1/e$ lần lời giải chính xác. Thuật toán xấp xỉ mà họ cung cấp cho hiệu quả tính toán các vị trí gần tối ưu với các đảm bảo hiệu suất lý thuyết mạnh mẽ. Phương pháp của Krause cho thấy các vị trí đặt trực quan với độ chính xác dự đoán vượt trội so với các phương pháp đương thời.

Sau này, nhiều tác giả cũng đã sử dụng bài toán này trong các thực nghiệm nghiên cứu về bài toán tối ưu hàm submodular và k-submodular [96, 114, 112, 106, 24].

1.3. Các vấn đề nghiên cứu có liên quan

Tối đa một hàm submodular là một bài toán NP-khó nhưng tồn tại một thuật toán xấp xỉ để giải bài toán đó [81]. Việc đề xuất các thuật toán xấp xỉ để giải đã thu hút rất nhiều sự quan tâm của các nhà nghiên cứu, từ các công bố của Nemhauser và cộng sự vào năm 1978 [101], cho tới các công bố trong 3 năm vừa qua [34, 67, 24, 41, 35, 44]... cho thấy đây là một chủ đề nóng và hiện đại.



Hình 1.5: Tìm vị trí đặt cảm biến tối đa thông tin thu được [81]

Các nghiên cứu về SM theo nhiều hướng khác nhau. Một số công bố nghiên cứu bài toán SM không có ràng buộc ($\mathcal{C} = \emptyset$) [55, 56, 20]... Các nghiên cứu được phát triển dựa trên công trình của Feige và cộng sự [55] khi xây dựng các thuật toán ngẫu nhiên để giải quyết bài toán này. Các tác giả chỉ ra tồn tại một thuật toán ngẫu nhiên xấp xỉ hiệu quả để với một hàm submodular không âm f và $f(\emptyset) = 0$, trả lại tập S sao cho tỉ lệ của lời giải so với lời giải tối ưu S^* thỏa mãn:

$$f(S) \geq \frac{1}{2} f(S^*).$$

Tuy nhiên, nghiên cứu về SM được tập trung nhiều hơn khi xét bài toán với ràng buộc. Việc đưa thêm ràng buộc vào cũng thỏa mãn nhiều kịch bản thực tế như thời gian, chi phí, số lượng... hầu hết sẽ bị giới hạn. Đối với SM có ràng buộc, đây vẫn là bài toán NP-khó [102], khả năng mở rộng của bài toán 1.1 bị hạn chế bởi độ khó của nó.

Vì vậy, các nhà nghiên thường tập trung vào đề xuất các thuật toán xấp xỉ hiệu quả với các đảm bảo lý thuyết về tỉ lệ xấp xỉ để giải quyết bài toán này. Dưới đây, luận án sẽ giới thiệu các nghiên cứu có liên quan với một số ràng buộc thường thấy của SM.

1.3.1. Bài toán SM với ràng buộc lực lượng

Ràng buộc lực lượng yêu cầu cần tìm tập lời giải $|S| \leq k$, với $k > 0$ là lực lượng lớn nhất của tập S. Khi đó bài toán trở thành *tối đa hàm submodular với ràng buộc lực lượng (Submodular Maximization under Cardinality constraint - SMC)*. Ứng dụng tiêu biểu của bài toán này có thể kể đến là bài toán *tối đa ảnh hưởng - IM)* đã mô tả ở trên.

Nemhauser và cộng sự [102, 101] là những người đầu tiên đề xuất thuật toán tham lam xấp xỉ cho bài toán SMC có hàm mục tiêu đơn điệu. Thuật toán của họ được chỉ ra tỉ lệ xấp xỉ là $1 - 1/e$ lời giải tối ưu.[4] Họ cũng chứng minh được rằng tính toán hàm f với không gian tập hợp là đa thức sẽ không thể cho tỉ lệ xấp xỉ tốt hơn $(1 - 1/e)$ [102, 101]. Sau này, các nhà khoa học tập trung giải bài toán SMC bằng các thuật toán cải tiến giảm thời gian chạy hoặc giảm dung lượng lưu trữ [97, 6, 11, 19]...

Mở rộng ràng buộc lực lượng, các thuật toán tham lam cũng được áp dụng cho nhiều ràng buộc phức tạp hơn như ràng buộc matroid [23, 50, 10], nhiều matroid (p-set constraint) [22, 66]... Bài toán trở thành *tối đa hàm submodular với ràng buộc matroid (submodular maximization under matroid constraint)*.

Bài toán này giả thiết cho $(V, \mathcal{I})$ là một matroid, với V là tập cơ sở, $\mathcal{I} \subseteq 2^V$ là một tập gồm các tập con độc lập lẫn nhau được tạo ra từ tập V. $\mathcal{I}$ thỏa mãn 2 tính chất: (1) với mọi $A \subseteq B \subseteq V$ và $B \in \mathcal{I}$ thì $A \in \mathcal{I}$; (2) $A, B \in \mathcal{I}$ và nếu $|B| > |A|$ thì $\exists e \in B \setminus A$ sao cho $A \cup \{e\} \in \mathcal{I}$. Hàm f gắn với matroid $(V, \mathcal{I})$ còn gọi là hàm tính hạng (rank function) của matroid. Bài toán cực đại hàm $f(\cdot)$ trong trường hợp này cho tỉ lệ xấp xỉ là $1/2$ khi sử dụng thuật toán tham lam. Cũng có nhiều tác giả đã đề xuất các phương pháp cải tiến khác nhằm tăng tốc thuật toán [23, 50, 10], hoặc giải với nhiều matroid $(V, \mathcal{I}_1), (V, \mathcal{I}_2), .., (V, \mathcal{I}_p)$ với $\mathcal{I} = \cap_i \mathcal{I}_i$, tạo nên ràng buộc p-matroid, cho tỉ lệ xấp xỉ là $1/(p + 1)$ [22].

1.3.2. Bài toán SM với ràng buộc chi phí

Một loại ràng buộc khác tổng quát hơn ràng buộc lực lượng là ràng buộc *chi phí (knapsack)*. Thay vì chỉ chọn k phần tử để thỏa mãn ràng buộc lực lượng, nhiều ứng dụng các phần tử được chọn phải thỏa mãn yêu cầu về ngân sách khi xét đến chi phí cần thiết để kích hoạt một phần tử và ngân sách để xét bài toán bị giới hạn. Khi đó, mỗi một phần tử $v \in V$ đều có một chi phí không đồng nhất $c(v) \geq 0$. Bài toán cho một ngân sách B, yêu cầu tìm lời giải S sao cho tối đa hàm submodular $f(S)$ với ràng buộc chi phí lời giải $c(S) \leq B$. Khi $c(e) = 1, \forall e \in V$, bài toán trở thành tối đa hàm submodular với ràng buộc lực lượng. Khi hàm mục tiêu f là tuyến tính, bài toán trở thành bài toán cái túi [38].

Thông thường hàm chi phí $c(\cdot)$ là một *hàm modular* hay *hàm có tính cộng tính* với chi phí của tập S bất kì sẽ là $c(S) = \sum_{v \in S} c(v)$. Bài toán lúc này trở thành *tối đa hàm submodular với ràng buộc chi phí (Submodular Maximization under Knapsack constraint - SMK)*. Hướng nghiên cứu với ràng buộc chi phí là hướng nghiên cứu chính của NCS trong luận án này.

[4] Một số bài báo viết thành $\frac{e}{e-1}$ do xét giữa tỉ lệ lời giải tối ưu với lời giải của thuật toán.

Bài toán SMK sở dĩ thu hút được nhiều sự quan tâm vì tính thách thức của nó. Thứ nhất, ràng buộc chi phí không giống như ràng buộc về lực lượng hay là matroid là những ràng buộc có thể dựa trên việc đếm, liệt kê các phần tử và chọn ra phần tử tốt nhất. Với bài toán SM, các thuật toán sẽ vừa phải tìm cực đại hàm f vừa phải đảm bảo tổng chi phí của lời giải không được vượt quá ngân sách cho trước. Vì thế, so với ràng buộc về lực lượng, ràng buộc chi phí sẽ tổng quát hơn. Nếu mỗi phần tử có chi phí là 1, bài toán SMK sẽ trở thành SMC.

Tiếp theo, vì phải quan tâm đến chi phí lời giải, bài toán SMK có thể cho rất nhiều lời giải với các chi phí khác nhau miễn không vượt quá ngân sách B. Do vậy kích thước lời giải cũng khác nhau. Vì thế, thời gian chạy của các thuật toán là khó xác định.

Ban đầu, Wolsey và cộng sự [146] khởi xướng nghiên cứu bài toán SMK, họ chứng minh được bài toán này là NP-khó nhưng có thể xấp xỉ lời giải với tỉ lệ $(1 - 1/e)$. Sau đó các tác giả khác [131, 92] đã đề xuất các phương pháp giải đạt hoặc gần đạt tỉ lệ $(1 - 1/e)$. Các tác giả khác như [88, 28, 21, 48] đã cố gắng cải tiến tỉ lệ xấp xỉ nói trên thành $(1 - 1/e + \epsilon)$ hoặc $(1/e + \epsilon)$, với $\epsilon > 0$ là tham số chính xác. Ngoài ra, họ còn mở rộng bài toán cho trường hợp tổng quát, hàm f có thể không đơn điệu [131].

Các thuật toán được đề xuất giải các bài toán trên thường được xây dựng dựa trên giải thuật tham lam [102, 146]. Tuy nhiên tối ưu hàm submodular là các bài toán NP-khó, nên các nghiên cứu sau này cải tiến thuật toán tham lam bằng thuật toán luồng [70, 62, 67] hoặc thuật toán song song [1, 67]... Gần đây, một số tác giả còn xem xét bài toán tối ưu hàm submodular với ràng buộc d nguồn ngân sách [148, 149], nhiều chi phí [130, 53]...

Vì tính thách thức và lợi ích của bài toán SMK, luận án tập trung nghiên cứu bài toán này và vận dụng nó trong một số trường hợp biến thể như có nhiễu, mở rộng thành k-submodular...

1.3.3. Bài toán Phủ Submodular

Các bài toán ứng dụng của SM như đặt cảm biến, tối đa ảnh hưởng... có thể dẫn về bài toán *Tập phủ (Set Cover)* [81, 80]. Trong đó, cần tìm tập phủ $S \subseteq V$ sao cho nó sẽ phủ tối đa các phần tử trong tập V. Tuy nhiên, có nhiều trường hợp người ta không cần quan tâm đến tối đa hàm mục tiêu mà chỉ cần vượt một ngưỡng cho trước là được, để chi phí bỏ ra là nhỏ nhất. Bài toán này còn gọi là *phủ tối thiểu*. Wolsey [145] đã khái quát thành *bài toán Phủ Submodular (Submodular Cover - SC)* có dạng như sau:

Định nghĩa 1.10 (Bài toán SC [145])**.** Cho một hàm submodular đơn điệu không

âm $f : 2^V \mapsto \mathbb{R}_+$ và một ngưỡng $\alpha > 0$, bài toán cần tìm một tập $S \subseteq V$ với lực lượng nhỏ nhất sao cho $f(S) \geq \alpha$.

Với bài toán này, chi phí về lực lượng nhỏ nhất là dạng chi phí đơn giản nhất. Tức là:

$$S^* \leftarrow \operatorname*{argmin}_S |S| \quad \text{s.t } f(S) \geq \alpha. \tag{1.6}$$

Dạng tổng quát của bài toán này bài toán *Tập phủ tối thiểu với chí phí nhỏ nhất (Min Cost Submodular Cover - MCSC)* trong đó mỗi đỉnh $e \in V$ sẽ có một chi phí $c(e) > 0$. Bài toán cần tìm tập S sao cho chi phí của tập này, $c(S) = \sum_{e \in S} c(e)$, là nhỏ nhất sao cho $f(S) \geq \alpha$.

Bài toán SC cũng đã được chứng minh là bài toán NP-khó bởi Wolsey [145]. Ông cũng đồng thời đưa ra một thuật toán tham lam cho tỉ lệ xấp xỉ là $1 - \ln(\max_{e \in V} f(e)/\beta)$ với β là lợi ích nhỏ nhất khác 0 của một phần tử nào đó được thêm vào tập lời giải bởi thuật toán. Goyal và cộng sự [63] sau đó đã đề xuất một thuật toán tham lam cho lời giải với chi phí bằng $1 - \log(\alpha/\epsilon)$ chi phí tối ưu và cho giá trị lời giải bằng $\alpha - \epsilon$ giá trị tối ưu.

Sau này, một số tác giả đã đưa các cải tiến vào trong thuật toán để giảm số lượng truy vấn xuống. Cụ thể, Crawford và cộng sự [40] đã đề xuất một thuật toán tiến hóa tiêu chí kép cho các tỉ lệ $(1 - \log(1/\epsilon), 1 - \epsilon)$ so với giá trị lời giải tối ưu và chi phí tối ưu. Gần đây, Ran và cộng sự [119] đã cải tiến chất lượng lời giải và số lượng truy vấn nhờ thiết kế thuật toán song song cho tỉ lệ xấp xỉ là $\frac{H(\min\{\max_{e \in S} g(e), \alpha\})}{1-5\epsilon}$ trong $O(n \log(n\alpha) \log(\alpha)(\log \alpha + \log\log(n\alpha))/\epsilon^4)$ truy vấn, với $H(\cdot)$ là một số Harmonic. Tuy nhiên, phương pháp của họ chỉ giải được với hàm f là số nguyên. Đồng thời, các thuật toán trên đây vẫn cho độ phức tạp truy vấn lớn. Một số tác giả còn mở rộng bài toán SC trên lưới nguyên $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$[127, 129] với lập luận rằng một phần tử có thể được lựa chọn nhiều lần thay vì một lần.

Với sự phát triển của bài toán SC trong nghiên cứu và ứng dụng, luận án đã mở rộng nghiên cứu với bài toán SC. Đặc biệt, luận án xem xét bài toán này trên lưới nguyên, trở thành bài toán DRSC để giải quyết các bài toán này.

1.3.4. Sự mở rộng của bài toán tối ưu hàm submodular

1.3.4.1. Mở rộng hàm mục tiêu thành k-submodular

Các công trình nghiên cứu và các ứng dụng của hàm submodular trong học máy và tối ưu tổ hợp cho thấy hàm này vừa có giá trị lý thuyết, vừa có ý nghĩa thực tiễn. Trong bối cảnh đó, tại một nghiên cứu của Lovász về hàm submodular [95], ông đã đặt vấn đề xét hàm submodular trong không gian nhiều chiều. Khi đó, tính chất của submodular có thêm nhiều điều mới mẻ mà được ông đánh giá là có sự bổ sung tính chất lý thuyết sâu sắc.

Các ứng dụng cụ thể của bài toán tối đa hàm k-submodular như tối đa ảnh hưởng k chủ đề, phân lớp, đặt k loại cảm biến... cho thấy việc nghiên cứu về tối ưu hàm k-submodular không chỉ có giá trị lý thuyết mà còn có giá trị thực tiễn. Quá trình nghiên cứu giải bài toán tổng quát bắt đầu khi Singh và cộng sự trình bày nghiên cứu về tối đa hàm k-submodular [125] với trường hợp $k = 2$. Kể từ thời điểm đó trở đi, nhiều công trình đã tập trung vào các bài toán xét k nói chung ($k \geq 2$), vì với $k = 1$ bài toán trở thành tối đa hàm submodular đã có.

Tối đa hàm submodular được nghiên cứu mở rộng thành tối đa hàm k-submodular với nhiều điều kiện khác nhau như tối đa hàm bi-submodular, k-submodular không ràng buộc [144, 135, 110], và có ràng buộc [109, 122, 151, 118]...

Ban đầu, các tác giả nghiên cứu bài toán tối đa hàm k-submodular không có ràng buộc bằng nhiều phương pháp khác nhau như: thiết kế thuật toán tham lam tất định [144], tham lam ngẫu nhiên [73], giải ngẫu nhiên [110], và thiết kế thuật toán luồng [126]... Tuy nhiên sau này, các nghiên cứu chuyển sang giải với ràng buộc nhiều hơn do có sự gần gũi với các bài toán xảy ra trong thực tế.

Đầu tiên, các nghiên cứu tập trung vào ràng buộc lực lượng. Oshaka và cộng sự [109] tiên phong giải bài toán tối đa hàm k-submodular đơn điệu với các ràng buộc về lực lượng bằng cách sử dụng thuật toán tham lam với tỉ lệ xấp xỉ $1/2$ cho ràng buộc kích thước tổng của cả tập và xấp xỉ $1/3$ cho ràng buộc kích thước đơn lẻ của từng tập con. Sau đó, các tác giả trong [151, 152] đã đề xuất nhiều phương pháp giải khác nhau, tuy nhiên tỉ lệ xấp xỉ tốt nhất đối với bài toán này vẫn là $1/2$ và độ phức tạp truy vấn chưa là tuyến tính.

Sau này, việc tối đa hàm k-submodular đã được nghiên cứu dưới các ràng buộc matroid. Các tác giả trong [122] đã chỉ ra một thuật toán tham lam có thể trả về tỉ lệ xấp xỉ là $1/2$ cho bài toán dưới ràng buộc matroid. Một thuật toán khác có cùng tỉ lệ xấp xỉ bằng cách sử dụng phương pháp tham lam liên tục riêng biệt đã được đề xuất trong [118]. Tuy nhiên, các phương pháp mở rộng hàm mục tiêu thành hàm liên tục cần các tính toán phức tạp và có thể dẫn đến các kết quả tùy ý

khó lường trước được [8].

Tiêu biểu, Ward và cộng sự [144] đã hoàn thiện lý thuyết nghiên cứu về hàm k-submodular. Thêm vào đó, một số tác giả khác [109, 116, 106]... đã chỉ ra các ứng dụng của bài toán tối đa hàm k-submodular trong nhiều tình huống, khi cần thu thập thông tin từ nhiều nguồn khác nhau. Từ đó hình thành nhiều bài toán như tóm tắt nhiều kiểu văn bản, lan truyền thông tin với nhiều chủ đề, tìm tập phủ cực đại cho nhiều loại cảm biến khác nhau... mà tối ưu hàm submodular chưa giải quyết được. Do vậy, việc nghiên cứu bài toán tối ưu hàm dạng k-submodular vừa có giá trị lý thuyết vừa có giá trị thực tiễn.

Hàm k-submodular được định nghĩa như sau:

Định nghĩa 1.11 (k-submodular [144])**.** Cho một tập cơ sở V và một số nguyên dương k, quy ước $[k] = \{1, 2, \ldots, k\}$ và $(k+1)^V = \{(V_1, V_2, \ldots, V_k) | V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}$ là họ k tập không giao nhau được gọi là *k-tập (k-set)*. Hàm $f : (k+1)^V \mapsto \mathbb{R}_+$ là *k-submodular* khi và chỉ khi với các biến $\mathbf{x}, \mathbf{y}$ bất kì $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ và $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, ta có:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}), \tag{1.7}$$

trong đó:

$$\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, \ldots, X_k \cap Y_k),$$

và

$$\mathbf{x} \sqcup \mathbf{y} = (Z_1, \ldots, Z_k), \text{ với } Z_i = X_i \cup Y_i \setminus (\bigcup_{j \neq i} X_j \cup Y_j).$$

Bài toán tối đa hàm k-submodular đã thu hút được nhiều sự quan tâm trong một vài năm gần đây. Các nghiên cứu chủ yếu tập trung vào tối đa hàm k-submodular có ràng buộc [109, 116, 126, 106, 152]...

1.3.4.2. Mở rộng hàm mục tiêu trên lưới nguyên

Trong các nghiên cứu ban đầu, thường xét bài toán tối ưu hàm submodular với hàm mục tiêu là hàm tập hợp. Tức là: $f : 2^V \mapsto \mathbb{R}_+$, là submodular nếu và chỉ nếu nó thỏa mãn tính chất *lợi nhuận hiệu suất giảm dần* (Định nghĩa 1.4).

Tuy nhiên, khi các ứng dụng được triển khai trên hàm tập hợp submodular chưa xét đến một tình huống thực tế, một phần tử tốt có thể được lựa chọn nhiều lần vào tập lời giải. Ví dụ khi một công ty chọn các đại lý tốt để tiếp thị một số lượng sản phẩm, công ty sẽ có xu hướng chọn đi chọn lại các đại lý mang lại lợi nhuận cao cho công ty. Như vậy việc chọn một đại lý như chọn một phần tử trong

tập hợp, việc chỉ chọn một lần để thêm vào tập lời giải sẽ chưa phản ánh hết được ý nghĩa đóng góp về mặt lợi ích của đại lý đó.

Các tình huống tương tự trên thúc đẩy các nhà nghiên cứu tìm hiểu các phiên bản tổng quát hóa của tính submodular và lợi nhuận hiệu suất giảm dần được định nghĩa trên *lưới nguyên (Integer lattice).* Soma và Yoshida [128] lần đầu tiên mở rộng hàm f trên lưới nguyên $\mathbb{Z}_+^V$. Các tác giả đã chỉ ra *tính chất lợi nhuận hiệu suất giảm dần trên lưới nguyên (Diminishing return submodular, thường gọi là DR-submodular)* trở thành phiên bản tổng quát của hàm submodular trên lưới nguyên.

Hàm DR-submodular được định nghĩa như sau:

Định nghĩa 1.12 (DR-submodular [128])**.** Cho một số nguyên dương $k \in \mathbb{N}$, ký hiệu $[k]$ đại diện cho tập $\{1, \ldots, k\}$. Cho tập cơ sở $V = \{e_1, \ldots, e_n\}$, ký hiệu $\mathbf{x}(e)$ là giá trị tọa độ của vec-tơ $\mathbf{x} \in \mathbb{Z}_+^V$ ứng với phần tử e nào đó. Ta cũng ký hiệu vec-tơ đơn vị thứ e là χ_e với $\chi_e(t) = 1$ nếu $t = e$ và $\chi_e(t) = 0$ nếu $t \neq e$. Với mọi vec-tơ $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, nói rằng $\mathbf{x} \leq \mathbf{y}$ nếu và chỉ nếu $\mathbf{x}(e) \leq \mathbf{y}(e), \forall e \in V$.

Hàm $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$ thỏa mãn tính chất DR-submodular khi và chỉ khi:

$$f(\mathbf{x} + \chi_e) - f(\mathbf{x}) \geq f(\mathbf{y} + \chi_e) - f(\mathbf{y}), \tag{1.8}$$

với $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V, \mathbf{x} \leq \mathbf{y}$.

Nếu f thỏa mãn tính chất 1.8 thì nó cũng thỏa mãn tính chất *lattice submodular* được định nghĩa như sau:

Định nghĩa 1.13 (Lattice submodular [128])**.** Nếu f là DR-submodular thì nó cũng thỏa mãn tính chất lattice submodular sao cho:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}), \tag{1.9}$$

trong đó, $\wedge$ và $\vee$ lần lượt là các phép toán nhỏ nhất, lớn nhất trên từng tọa độ. Có nghĩa là: $\mathbf{x} \wedge \mathbf{y}(e) = \min\{\mathbf{x}(e), \mathbf{y}(e)\}$, và $\mathbf{x} \vee \mathbf{y}(e) = \max\{\mathbf{x}(e), \mathbf{y}(e)\}$.

Khi hàm submodular là hàm tập hợp, tính chất submodular và tính chất lợi nhuận hiệu suất giảm dần là tương đương. Nhưng xét trên lưới nguyên, tính chất submodular đã được phát triển thành tính chất DR-submodular (1.8) và lattice submodular. Tính chất DR-submodular mạnh hơn lattice submodular (1.9). Có phát biểu rằng f là lattice submodular nếu f thỏa mãn bất đẳng thức (1.9), và nếu f tiếp tục thỏa mãn bất đẳng thức (1.8), f là DR-submodular [128].

DR-submodular là sự tổng quát hóa tự nhiên nhất của submodular và có ý nghĩa trong nghiên cứu do nó có liên quan đến tính toán lợi ích thu được. Vì thế

đối với các nghiên cứu gần đây [127, 128, 129] người ta đã đặt vấn đề nghiên cứu hàm DR-submodular ứng dụng trong các bài toán tối ưu phân bố tài nguyên và bài toán phúc lợi xã hội. Do tính mới của nó, luận án cũng đặt mục tiêu nghiên cứu bài toán tối ưu với hàm mục tiêu là DR-submodular.

1.4. Thuật toán xấp xỉ giải quyết bài toán tối ưu hàm submodular

1.4.1. Khái niệm thuật toán xấp xỉ

Thuật toán xấp xỉ là các thuật toán tìm lời giải gần đúng cho các bài toán tối ưu. Thuật toán xấp xỉ thường được sử dụng cho các bài toán NP-khó, hoặc các bài toán có thuật toán đa thức nhưng quá chậm khi dữ liệu đầu vào lớn.

Với một bài toán tối ưu, mỗi lời giải tiềm năng đều có một chi phí dương nào đó, và ta cần tìm lời giải gần tối ưu. Từ góc độ tính tỉ lệ xấp xỉ của các lời giải cho thấy các bài toán NP-khó có độ khó rất khác nhau.

Chúng ta biểu thị một thuật toán có *tỉ lệ xấp xỉ (approximation ratio)* là $\rho(n)$ với n là kích thước dữ liệu đầu vào, thường được viết gọn lại là ρ, nếu với bất kì một tập đầu vào kích thước n, chi phí C của lời giải là không quá hệ số ρ của chi phí C^* của lời giải tối ưu [138].

Định nghĩa cụ thể hơn, với bài toán CO, giả sử có hàm mục tiêu $f : 2^V \mapsto \mathbb{R}$ với V là tập cơ sở kích thước n. Gọi $S^* \subseteq V$ là *lời giải tối ưu của bài toán (optimal solution)*, giá trị $f(S^*)$ tương ứng là *giá trị tối ưu của bài toán (optimal value)*, ký hiệu là opt. Khi đó, định nghĩa thuật toán xấp xỉ của bài toán CO như sau:

Định nghĩa 1.14 (Thuật toán xấp xỉ [138])**.** Giả sử cần tìm lời giải $S \subseteq V$ với V là tập cơ sở kích thước n sao cho hàm f đạt giá trị cực đại. Ta nói thuật toán xấp xỉ $\mathcal{A}$ cho lời giải là $S \subseteq V$ có tỉ lệ xấp xỉ là $\rho, \rho \in (0, 1)$, nếu nó thực hiện trong thời gian đa thức theo kích cỡ của dữ liệu đầu vào và thỏa mãn

$$f(S) \geq \rho \cdot \mathsf{opt}. \tag{1.10}$$

Với bài toán tìm giá trị cực tiểu của hàm f, một cách tương tự sẽ có: $f(S) \leq \rho.\mathsf{opt}, \rho > 1$. ρ được gọi là *tỉ lệ xấp xỉ hoặc hệ số xấp xỉ.*

1.4.2. Các đảm bảo lý thuyết của thuật toán

1.4.2.1. Tỉ lệ xấp xỉ

a) Tỉ lệ xấp xỉ đảm bảo chất lượng lời giải

Tỷ lệ xấp xỉ ρ có ý nghĩa rằng trong trường hợp xấu nhất, thuật toán cũng đảm bảo đạt tỉ lệ ρ lần lời giải tối ưu. Giá trị này rất quan trọng trong việc thiết

kế các thuật toán xấp xỉ, thường được sử dụng để so sánh đánh giá giữa các thuật toán khi cùng giải một bài toán.

Với thuật toán xấp xỉ tỉ lệ 1 thì đây là lời giải tối ưu, và nếu tỉ lệ theo công thức (1.10) càng bé thì lời giải các tồi. Nhiều thuật toán giải bài toán NP sẽ cho tỉ lệ xấp xỉ là một hằng số (*constant-factor approximation*). Chẳng hạn thuật toán tham lam giải bài toán SM cho tỉ lệ xấp xỉ là $1 - 1/e \approx 63\%$ là một hằng số [102].

Bài toán NP gây khó khăn lớn khi tỉ lệ xấp xỉ bị ảnh hưởng bởi kích cỡ n. Vì thế, một trong các hướng thiết kế thuật toán là đề xuất các *thuật toán xấp xỉ tất định (deterministic approximation algorithm)* với tỉ lệ là hằng số để tăng sự độc lập và tính xác định của thuật toán với bất kì kích cỡ n nào của dữ liệu đầu vào. Thuật toán tất định là những thuật toán có thể dự đoán được đầu ra dựa trên đầu vào của nó. Nói cách khác, trong một thuật toán tất định, đối với một đầu vào cụ thể nhất định, máy tính sẽ luôn tạo ra cùng một đầu ra thông qua các trạng thái giống nhau.

Ngoài ra, khi đưa ra tỉ lệ xấp xỉ hoặc độ phức tạp, thường xuất hiện tham số $\epsilon > 0, \epsilon \in (0, 1)$. Tham số này gọi là *tham số chính xác (accuracy parameter)* [138]. Chẳng hạn như C.Borgs và cộng sự để giải bài toán IM, đã đề xuất cải tiến thuật toán tham lam bằng phương pháp lấy mẫu ảnh hưởng ngược để tìm tập lời giải. Thuật toán của họ cho tỉ lệ xấp xỉ là $1 - 1/e - \epsilon$ với độ phức tạp thời gian giảm theo hệ số logarit so với thuật toán đã được đề xuất bởi Kempe [43].

Lưu ý trong một số công bố, các tác giả đưa ra tỉ lệ xấp xỉ là $\rho > 1$, khi đó tỉ lệ xấp xỉ căn cứ giữa tỉ lệ lời giải tối ưu hơn lời giải xấp xỉ nhiều nhất là bao nhiêu lần. Chẳng hạn trong các bài báo [1, 67], các tác giả đã đề xuất các thuật toán xấp xỉ giải bài toán SMK với các tỉ lệ xấp xỉ là $6 + \epsilon$, $5 + \epsilon$ và $4 + \epsilon$. Với cách trình bày này, tỉ lệ xấp xỉ càng gần giá trị 1 thì lời giải càng tốt.

b) Thuật toán xấp xỉ tiêu chí kép

Khi nghiên cứu các bài toán cần tối ưu chi phí, việc đưa ra lời giải cho tỉ lệ xấp xỉ là hằng số hoặc đề xuất các thuật toán xấp xỉ đưa ra lời giải trong thời gian đa thức sẽ gặp nhiều thách thức do chịu sự ràng buộc của chi phí lời giải tối ưu phải nhỏ nhất trong khi hàm lợi ích (hàm mục tiêu) vẫn phải vượt ngưỡng. Khi đó, một số tác giả đã đề xuất hướng tiếp cận nới lỏng các ràng buộc đối với cả hàm chi phí và hàm mục tiêu. Cách thiết kế theo hướng tiếp cận như vậy gọi là thiết kế *thuật toán xấp xỉ tiêu chí kép (bicriteria approximation algorithm*).

Một thuật toán gọi là xấp xỉ tiêu chí kép (σ_1, σ_2) là thuật toán cho 2 tỉ lệ xấp xỉ σ_1, σ_2 lần lượt là các tỉ lệ xấp xỉ của giá trị chi phí và giá trị mục tiêu của lời giải có được từ thuật toán so với lời giải tối ưu. Ví dụ, Crawford và cộng sự [**?**] đã

đề xuất một thuật toán tiến hóa tiêu chí kép cho các tỉ lệ $(1 - \log(1/\epsilon), 1 - \epsilon)$ so với chi phí tối ưu và giá trị lời giải tối ưu cho bài toán tối thiểu chi phí cho SC.

Tỉ lệ xấp xỉ thể hiện chất lượng lời giải so với lời giải tối ưu. Tuy nhiên, một thuật toán cho chất lượng lời giải tiệm cận với lời giải tối ưu nhưng thời gian chạy của nó quá lớn làm cho nó trở nên bất khả thi. Vì thế, khi đề xuất các thuật toán xấp xỉ cần phải quan tâm đến các đảm bảo lý thuyết khác của thuật toán như thời gian chạy, dung lượng bộ nhớ... đặc biệt là thời gian chạy, được thể hiện qua độ phức tạp của thuật toán.

1.4.2.2. Độ phức tạp thuật toán

Trong các đặc trưng của thuật toán, thì khối lượng công việc cần thực hiện là một đặc trưng thiết yếu. Đây là tiêu chí quan trọng nhất để đánh giá một thuật toán có tốt hay không. Khối lượng công việc thường tỉ lệ thuận với thời gian thực hiện thuật toán. Từ đó, khái niệm *độ phức tạp (thời gian)* được quan tâm. Đối với bài toán tối ưu hàm submodular, một vài độ phức tạp ảnh hưởng tới kết quả chạy của thuật toán được chỉ ra như dưới đây.

a) Độ phức tạp thời gian

Hiện nay, các bài toán bùng nổ với dữ liệu tăng nhanh về kích cỡ nên việc thiết kế một một thuật toán hiệu quả cũng rất quan trọng. Rõ ràng thuật toán giải quyết bài toán trong vài giờ sẽ được đánh giá cao hơn một thuật toán khác phải mất vài ngày, với điều kiện độ chính xác của các thuật toán là tương đương nhau. Khi đánh giá thuật toán hoặc so sánh giữa các thuật toán, người ta thường đưa ra phép đo thời gian thực hiện thuật toán, hay còn gọi là *độ phức tạp thời gian (time complexity)*, đôi khi sử dụng thuật ngữ độ phức tạp tính toán cũng mang ngụ ý về độ phức tạp thời gian.

Gọi độ phức tạp thời gian là một hàm $T(n)$ phụ thuộc vào kích thước dữ liệu đầu vào $n \in \mathbb{Z}_+$. Với tập hợp, kích thước dữ liệu là số phần tử của tập đó. Độ phức tạp thời gian được xem xét trong các trường hợp: xấu nhất, tốt nhất, hoặc trong trường hợp trung bình [38]...

Thông thường, độ phức tạp xấu nhất (hoặc tồi nhất) được sử dụng thường xuyên hơn cả để đánh giá độ phức tạp thuật toán, vì nó đánh giá khả năng xấu nhất sẽ xảy ra của một thuật toán, để người sử dụng có thể ước lượng được thời gian hoặc chi phí cần thiết để thực hiện thuật toán.

Định nghĩa 1.15 (Độ phức tạp xấu nhất [38])**.** Độ phức tạp xấu nhất là thời gian lâu nhất để thực hiện thuật toán. Trường hợp này lượng dữ liệu vào là bất lợi nhất.

Cụm từ *độ phức tạp* hay *độ phức tạp thời gian* cũng mang hàm ý độ phức tạp

xấu nhất. Thời gian chạy tồi nhất của thuật toán là cận trên cho bất cứ đầu vào nào. Ngoài ra, độ phức tạp tồi nhất cũng rất thường xuyên xảy ra trong thực tế. Chẳng hạn, tìm kiếm một thông tin nào đó trong kho dữ liệu, mà thông tin này không có là điều thường gặp.

Để làm căn cứ đánh giá một cách tổng quát các thuật toán, người ta thường *xấp xỉ thời gian chạy của thuật toán*. Trong đó, hàm O, còn gọi là O-lớn (*Big-oh*) cho một xấp xỉ của thời gian chạy của thuật toán trong trường hợp xấu nhất.

Định nghĩa 1.16 (O-lớn [4])**.** Gọi $T(n)$ là thời gian chạy thuật toán và cho hàm $f(n)$, ta nói $T(n)$ là một O- lớn, $O(f(n))$, nếu tồn tại hằng số thực C và số nguyên dương n_0 sao cho

$$T(n) \leq C \cdot f(n) \, \forall n \geq n_0. \tag{1.11}$$

Có thể viết $T(n) = O(f(n))$, hay thuật toán có độ phức tạp là $O(f(n))$. Ví dụ $T(n) = 5n^2 + 2n \leq 7n^2$ là hàm mô tả phức tạp (thời gian chạy) của một thuật toán nào đó, vậy $T(n) = O(n^2)$, với $C = 7, n_0 = 1$, hay thuật toán có độ phức tạp là $O(n^2)$.

Một số độ phức tạp tính toán thường gặp cùng với tên gọi của nó được trình bày trong Bảng 1.1.

Bảng 1.1: Bảng độ phức tạp tính toán

Ký hiệu	Tên gọi độ phức tạp
$O(1)$	Hằng số (*constant*)
$O(\log n)$	Lô-ga-rít (*logarithmic*)
$O(n)$	Tuyến tính (*linear*)
$O(n \log n)$	Gần tuyến tính (*linearithmic*|*sublinear*)
$O(\log \log n)$	Lô-ga-rít kép (*double logarithmic*)
$O(n^2)$	Bình phương (*quadratic*)
$O(n^c)$	Đa thức (*polynomial*)
$O(c^n), c > 1$	Hàm mũ (*exponential*)

$O(\cdot)$ cũng được coi là *chặn trên (upper bound)* của thuật toán và chặn trên của một thuật toán là lượng thời gian cần thiết nhiều nhất (hiệu suất trong trường hợp xấu nhất). Ký hiệu $O(\cdot)$ được sử dụng để mô tả cận trên tiệm cận mang hàm ý số phép tính nhiều nhất cần dùng. Ngược lại, hàm Ω - còn gọi là Ô-mê-ga lớn cho ước lượng *chặn dưới (lower bound)* của số phép toán cần dùng.

Định nghĩa 1.17 (Ô-mê-ga lớn [4])**.** Cho hàm số $f(n)$, ta nói hàm $T(n)$ là một $\Omega(f(n))$, đọc là Ô-mê-ga lớn của $f(n)$, nếu tồn tại số dương C và số nguyên n_0, sao cho

$$T(n) \geq C \cdot f(n), \forall n \geq n_0. \tag{1.12}$$

Giả sử $T(n) = 3n^2 + 5n - 4$, ta có $T(n) = \Omega(n^2)$, hoặc $T(n) = \Omega(n)$ hoặc thậm chí $T(n) = \Omega(1)$.

Trong trường hợp muốn xấp xỉ hàm thời gian $T(n)$ theo trung bình, hoặc cả hai tiêu chuẩn O và Ω, người ta đưa ra thêm một tiêu chuẩn nữa cho phép ước lượng thời gian chạy của thuật toán trong hai trường hợp này là Θ. Hàm này được định nghĩa như sau:

Định nghĩa 1.18 (Thê-ta lớn [38])**.** Cho hàm số $g(n)$, ta nói hàm $T(n)$ là một $\Theta(g(n))$, đọc là Thê-ta lớn của $g(n)$, nếu tồn tại hai số dương C_1, C_2 và số nguyên n_0, sao cho

$$0 \leq C_1 g(n) \leq T(n) \leq C_2 \cdot g(n), \forall n \geq n_0 \tag{1.13}$$

Giả sử $T(n) = 3n^2 + 25n + 4$, có thể viết $T(n) = \Theta(n^2)$. Thê-ta lớn là giới hạn chặt nhất mà thuật toán có thể thực hiện. Tuy nhiên, hiện nay độ phức tạp $O(\cdot)$ được sử dụng nhiều hơn cả do có sự liên quan đến thời gian lâu nhất cần ước tính để chạy thuật toán.

b) Độ phức tạp truy vấn

Đối với bài toán tối ưu hàm submodular, bên cạnh quan tâm đến độ phức tạp thời gian, người ta còn quan tâm đến độ phức tạp truy vấn (*query complexity*). Một lần thực hiện ước lượng hàm mục tiêu f được gọi là một *truy vấn (query)*. Một query là một truy vấn đa thức theo thời gian. Với độ phức tạp truy vấn, độ phức tạp O-lớn thường được sử dụng. Trong các công bố về hàm submodular gần đây [1, 67, 41, 44, 35]... để so sánh giữa các thuật toán, các tác giả thường đưa ra độ phức tạp truy vấn như là một đại diện cho thời gian chạy.

Định nghĩa 1.19 (Độ phức tạp truy vấn)**.** Giả sử bài toán đã cho một cách ước lượng hàm mục tiêu f, coi một lời gọi hàm là một phép tính đa thức theo thời gian, một truy vấn là một lời gọi hàm f. Độ phức tạp truy vấn là số lượng truy vấn nhiều nhất mà thuật toán cần thực hiện.

Với các bài toán CO nói chung và các bài toán tối ưu hàm submodular nói riêng, việc gọi hàm f là thường xuyên, nên số lượng lời gọi hàm ảnh hưởng trực

tiếp đến thời gian chạy của thuật toán. Do vậy đây là một phép đo quan trọng để xác định hiệu quả của một thuật toán. Các thuật toán thiết kế theo hướng giảm độ phức tạp truy vấn đang là xu hướng chung khi giải bài toán tối ưu hàm submodular.

Với thuật toán tham lam cổ điển cho bài toán SMC [101], cần $O(nk)$ truy vấn tới hàm f. Ngoài ra, với một số bài toán, ví dụ tối đa hàm submodular không ràng buộc, tối đa hàm submodular với ràng buộc chi phí (bài toán SMK)... độ lớn của tập lời giải không xác định và phụ thuộc chặt chẽ vào tập dữ liệu đầu vào cùng với các tham số đầu vào khác. Khi đó, số lượng lời giải dự tuyển sẽ khó lường trước được và các lời giải sẽ có kích cỡ khác nhau. Vì vậy, để đánh giá một thuật toán theo thời gian chạy là khó khả thi. Thay vào đó, người ta đánh giá thông qua độ phức tạp truy vấn. Cho nên sử dụng độ phức tạp truy vấn là một tiêu chí quan trọng để đánh giá, so sánh giữa các thuật toán.

1.4.3. Thuật toán tham lam xấp xỉ

Thuật toán tham lam xấp xỉ là một thuật toán hiệu quả để giải quyết các bài toán tối ưu hàm submodular. Nemhauser và cộng sự [102] lần đầu tiên xây dựng một giải thuật tham lam để giải bài toán SMC với hàm mục tiêu là hàm submodular không âm đơn điệu tăng. Các tác giả xây dựng chiến lược tham lam duyệt tuần tự từng phần tử trong tập cơ sở và chọn ra các phần tử nào cho lợi nhuận biên lớn nhất, cho tới khi điều kiện dừng được thoả mãn (số lượng phần tử của tập lời giải đạt đến k). Chi tiết của giải thuật được trình bày trong Thuật toán 1

Algorithm 1 : Greedy algorithm

Input: $f : 2^V \mapsto \mathbb{R}^+$; $f(\emptyset) = 0$, $k \in \mathbb{Z}_+$
Output: S

1: $S \leftarrow \emptyset, V^0 = V, t = 1$;
2: **while** $t \leq k$ **do**
3: $\quad e_{max} \leftarrow \arg\max_{e \in V \setminus S}(f(S + \{e\}) - f(S))$
4: $\quad S \leftarrow S \cup \{e_{max}\}$
5: $\quad t \leftarrow t + 1$
6: **end while**
7: **return** S

Nemhauser và Wolsey [101] cũng đã chỉ ra sự phù hợp của thuật toán tham lam với bài toán SM khi cho tỉ lệ xấp xỉ tốt nhất. Hơn nữa, sử dụng phương pháp tham lam còn thể hiện sự linh hoạt khi áp dụng cho nhiều dạng của bài toán tối ưu hàm submodular. Chẳng hạn với bài toán SMK, thuật toán tham lam do Wolsey đề xuất [146] lựa chọn các phần tử cho tỉ lệ giữa lợi nhuận biên và chi phí là lớn nhất

$e_{max} \leftarrow \arg\max_{e \in V \setminus S}(f(S \cup e) - f(S))/c(e)$. Thuật toán này cũng cho tỉ lệ xấp xỉ $1 - 1/e$.

Nhược điểm của giải thuật tham lam đó là số phép tính cần thực hiện rất lớn. Thuật toán chạy tuần tự, tại mỗi bước lặp của t cần tốn $O(n)$ truy vấn hàm f. Như vậy, Thuật toán 1 có độ phức tạp thời gian là $O(nk)$, độ phức tạp truy vấn là $O(nk)$, độ phức tạp bộ nhớ là $O(n)$. Điều này làm thuật toán trở nên bất khả thi với bộ dữ liệu có kích cỡ lớn. Chương 2 của luận án cũng đã chỉ ra trong phần thực nghiệm đối với bài toán kSMK, với các bộ dữ liệu vài chục nghìn đỉnh, với thời gian bị ràng buộc, thuật toán tham lam không thể cho lời giải cuối cùng trong khi các thuật toán cải tiến đều trả về kết quả.

1.4.4. Cải tiến thuật toán tham lam xấp xỉ

Với bài toán tối đa hàm submodular giải quyết bằng giải thuật tham lam đơn giản [102, 145, 146], thuật toán phải xét tuần tự tất cả các phần tử trong tập cơ sở V để tìm ra phần tử có đóng góp lợi ích lớn nhất. Với cách làm này thời gian tìm kiếm sẽ bị chi phối bởi số lời gọi hàm f. Kết hợp với dữ liệu đầu vào lớn, thời gian chạy của thuật toán tham lam sẽ tăng nhanh do không gian tìm kiếm tăng theo hàm mũ. Điều đó dẫn đến yêu cầu cần phải thiết kế các thuật toán nhanh với số lượng truy vấn lời gọi hàm giảm xuống đáng kể so với tham lam.

Hơn nữa, còn một yêu cầu phát sinh từ thực tiễn nghiên cứu với các bài toán ứng dụng cụ thể của SM, Amanatidis và đồng sự [2] đã chỉ ra thách thức gặp phải khi có sự không chắc chắn cố hữu trong các vấn đề như vị trí cảm biến hoặc tối đa hóa doanh thu, trong đó người ta không tìm hiểu giá trị đóng góp lợi ích chính xác của một phần tử cho đến khi phần tử đó được thêm vào giải pháp. Và vì vậy, thuật toán cũng sẽ phải trả giá cho việc tính toán hàm mục tiêu. Ngay cả việc ước tính giá trị kỳ vọng cho một hàm mục tiêu chưa biết một phần nào đó cũng có thể rất tốn kém. Do đó việc giảm số lời gọi hàm càng trở nên cần thiết.

Để khắc phục vấn đề thời gian chạy của thuật toán tham lam, các nhà khoa học thường đưa ra các giải pháp cải tiến theo hướng giảm độ phức tạp thời gian, giảm độ phức tạp truy vấn, hoặc giảm độ phức tạp bộ nhớ nhưng vẫn đảm bảo tỉ lệ xấp xỉ chấp nhận được. Một số phương pháp cải tiến phổ biến được tóm tắt như sau.

1. **Thiết kế ngưỡng tham lam**. Thuật toán tham lam xấp xỉ cải tiến đưa ra một *ngưỡng tham lam (greedy threshold)* θ để có thể chọn ra các phần tử "tốt" và loại bỏ các phần tử "xấu".

Phương pháp ngưỡng tham lam đề ra một tiêu chí nào đó, sử dụng một ngưỡng θ theo tiêu chí đó để lọc các phần tử. Căn cứ vào ngưỡng θ này, thuật toán giữ lại các phần tử thoả mãn tiêu chí và loại bỏ các phần tử không vượt qua được điều

kiện ngưỡng. Đây là một cách thiết kế tốt để giảm số lượng truy vấn hàm mục tiêu.

Có nhiều cách để đề xuất ngưỡng θ này. Cách đầu tiên là *thiết kế ngưỡng tham lam cho lợi ích tăng thêm.* Phương pháp này yêu cầu giá trị đóng góp của 1 phần tử e vào trong tập lời giải dự tuyển S phải vượt qua một ngưỡng θ cho trước. Chẳng hạn với thuật toán SIEVE-STREAMING do Bandanidiyuru và cộng sự đề xuất [5] giải quyết bài toán SMC, ngưỡng θ thiết lập là giá trị lợi nhuận biên $\Delta(e|S_v)$ của e khi thêm vào tập lời giải dự tuyển S_v không nhỏ hơn $\theta = (v/2 - f(S_v))/(k - |S_v|)$ với v là một ước lượng của giá trị mục tiêu có thể tính toán được. Phương pháp này còn được áp dụng trong nhiều công bố sau này [86, 87, 34].

Một cách thiết kế khác là *thiết kế ngưỡng tham lam theo mật độ tăng thêm (density gain).* Phương pháp này dùng khi giải bài toán tối ưu hàm submodular với ràng buộc chi phí, ví dụ bài toán SMK. Wolsey lần đầu tiên đưa ra khái niệm *mật độ tham lam (density greedy)* khi giải quyết bài toán SMK đơn điệu [146]. Khái niệm mật độ là *tỉ lệ giữa lợi nhuận biên và chi phí của một phần tử* khi thêm vào tập lời giải.

Sau đó, Mirzasoleiman và cộng sự [98] lần đầu tiên xây dựng phương pháp *ngưỡng mật độ tham lam (Greedy Density Threshold - GDT).* Ý tưởng chính là xây dựng thủ tục GDT không chọn các phần tử nào có tỉ lệ $\Delta(e|S)/c(e)$ nhỏ hơn ngưỡng ρ với $c(e)$ là tổng ngân sách chi cho phần tử e từ nguồn l ngân sách. Điểm đặc biệt là các tác giả đã thiết kế để ρ có thể giảm dần theo từng vòng lặp nhằm gia tăng chất lượng lời giải, cải thiện tỉ lệ xấp xỉ của thuật toán. Lúc đó, phương pháp này được gọi là *ngưỡng tham lam giảm dần (decreasing greedy threshold).*

Sử dụng ngưỡng tham lam còn có ưu điểm là sự linh hoạt khi kết hợp với các phương pháp thiết kế khác như giảm dần ngưỡng, thiết kế thuật toán luồng, song song hoá để giảm độ phức tạp thời gian, độ phức tạp truy vấn mà vẫn đảm bảo chất lượng lời giải [2, 67, 24].

2. **Thiết kế thuật toán luồng**. Thiết kế thuật toán luồng nhằm gia tăng cơ hội chọn nhiều phần tử tốt với dung lượng bộ nhớ bị giới hạn.

Thông thường, với tham lam, toàn bộ dữ liệu được đưa vào thuật toán thành một luồng và được đọc tuần tự từng phần tử. Sau đó, tại một thời điểm cần phải đưa ra quyết định có chọn nó hay không. Rất nhiều trường hợp không thể đảo ngược được quyết định làm cho thời gian tìm ra lời giải lâu hơn. Đồng thời, dung lượng bộ nhớ luôn luôn phải chứa cả tập V. Việc xử lý tuần tự từng phần tử làm cho hiệu quả sử dụng bộ nhớ bị giảm đi.

Thuật toán luồng (streaming) là một dạng thuật toán trực tuyến được vận dụng nhằm cải tiến thuật toán tham lam. Thay vì đọc vào bộ nhớ toàn bộ luồng, thuật toán trực tuyến sẽ đọc dữ liệu vào theo từng khối hoặc từng phần. Với thuật toán

luồng, nó sẽ quét luồng dữ liệu có kích thước n một hoặc nhiều lần. Tại một thời điểm t, thuật toán sẽ đọc và xử lý các phần tử có kích thước hữu hạn M sao cho $M \ll n^t$ với n^t là kích thước của luồng dữ liệu. Để đánh giá hiệu quả của một thuật toán luồng, các căn cứ sau được đưa ra:

- Số k lần quét dòng dữ liệu (k-pass) mà thuật toán cần.
- Dung lượng bộ nhớ cần xử lý trong luồng, $\max |M_t|$;
- Thời gian chạy của thuật toán, thường được đo qua độ phức tạp truy vấn;
- Tỉ lệ xấp xỉ mà thuật toán đưa ra.

Bandanidiyuru và cộng sự [5] là những người tiên phong đưa kỹ thuật luồng kết hợp với tham lam giải bài toán SMC với các đảm bảo lý thuyết mạnh mẽ. Họ đã đề xuất thuật toán SIEVE-STREAMING chỉ cần duyệt dòng dữ liệu 1 lần, với sắp xếp bất kì, cho tỉ lệ xấp xỉ là $1/2 - \epsilon, \epsilon > 0$. Tại cùng 1 thời điểm, thuật toán này chỉ yêu cầu có $O((k \log k)/\epsilon)$ bộ nhớ. Tức là nó độc lập với kích thước của tập dữ liệu đầu vào, và xử lý các dữ liệu với $O((\log k)/\epsilon)$ thời gian cập nhật. Tỉ lệ xấp xỉ so với tham lam thì vẫn thấp hơn nhưng tính hiệu quả được đánh giá cao khi chỉ bằng 1 lần quét tập V, dung lượng nhớ và thời gian chạy đã được giảm đáng kể.

Ưu điểm của thuật toán luồng là có thể kết hợp với ngưỡng tham lam để sàng lọc các phần chọn ra phần tử tốt. Ngoài ra, khi quét dòng dữ liệu vài lần, đưa ra nhiều lời giải dự tuyển để cùng sàng lọc một phần tử tại một thời điểm tạo cơ hội chọn lại các phần tử tốt đã bị bỏ qua là lớn hơn. Qua đó đảm bảo được chất lượng lời giải cân bằng với số lượng truy vấn và dung lượng lưu trữ dữ liệu bị giới hạn.

Thiết kế thuật toán luồng hiện nay rất được ưa chuộng, áp dụng rộng rãi với nhiều dạng của bài toán tối ưu hàm submodular [115, 1, 67, 24]...

3. **Thiết kế thuật toán song song.** Thiết kế thuật toán song song dựa trên ý tưởng song song hoá các truy vấn lời gọi hàm tại mỗi bước lặp của thuật toán, do Balkanski và cộng sự [11] đề xuất. Từ đó hình thành khái niệm độ phức tạp song song là số vòng lặp tuần tự tối thiểu của một thuật toán. Song song hoá làm cho tại mỗi vòng lặp, thuật toán sẽ thực hiện các truy vấn đến hàm $f(\cdot)$ là độc lập lẫn nhau, do đó có thể tiến hành song song với nhau được. Điểm khó khăn lớn nhất của thuật toán song song đó là phụ thuộc vào người thiết kế thuật toán, rất khó để đưa ra một khuôn mẫu chung cho tất cả các bài toán tối ưu khác nhau được. Nhưng lợi điểm nhìn thấy rõ là nếu độ phức tạp song song thấp, tức là số vòng lặp tuần tự càng ít thì tốc độ chạy của thuật toán càng nhanh. Phương pháp này được áp dụng trong một số công bố gần đây [10, 2]...

1.5. Các thách thức và mục tiêu nghiên cứu

Bài toán tối ưu hàm submodular là một bài toán vừa có nhiều tính chất lý thuyết sâu sắc, vừa có tính ứng dụng trong thực tiễn cao. Nghiên cứu về bài toán này là một chủ đề nóng trong thời gian gần đây, xuất hiện nhiều trên các diễn đàn nổi tiếng về trí tuệ nhân tạo và học máy như các Hội nghị AAAI [86], AISTATS[85], ICML [96, 87, 106], NEURIPS [128, 2], IJCAI [120, 24]... cũng như các tạp chí hàng đầu về tối ưu tổ hợp và vận trù học [145, 131, 129]...

Qua tìm hiểu các công bố trên, tác giả nhận thấy xu hướng nghiên cứu và ứng dụng trong thực tiễn mà giới nghiên cứu đang chú trọng là các bài toán biến thể và mở rộng của tối đa hàm submodular. Các bài toán mới xét với nhiều loại ràng buộc, mở rộng không gian tìm kiếm... Tuy nhiên, hướng nghiên cứu này có một số thách thức như sau:

1. Các bài toán thuộc lớp bài toán NP-khó, cần các thuật toán xấp xỉ hiệu quả giải bài toán này. Vì thế, yêu cầu đặt ra đối với luận án là đề xuất được các thuật toán xấp xỉ cạnh tranh với các thuật toán hiện có.

2. Vấn đề về dữ liệu lớn. Sự bùng nổ của tập dữ liệu đầu vào, không gian tìm kiếm lời giải tăng theo hàm mũ. Do vậy, các thuật toán cần phải đảm bảo thời gian tìm kiếm nhanh và không gian lưu trữ tối ưu.

3. Giảm số lượng truy vấn. Việc tính toán hàm mục tiêu ảnh hưởng tới thời gian chạy của thuật toán. Với bài toán tối đa hàm submodular, lời gọi hàm mục tiêu được thực hiện nhiều lần. Do vậy, nảy sinh yêu cầu cần thiết kế các thuật toán sao cho giảm thiểu đáng kể các truy vấn này. Đây cũng là một thách thức quan trọng cần lưu ý khi thiết kế các thuật toán.

4. Tính chất không đơn điệu của hàm mục tiêu. Thực tế, không phải lúc nào hàm mục tiêu submodular cũng có thể đơn điệu. Tính không đơn điệu ảnh hưởng đến các thuật toán khi cần nhanh chóng phải loại bỏ các phần tử "xấu". Do vậy, các thuật toán cần phải có các phân tích và đưa ra cách chọn lời giải khéo léo.

5. Cuối cùng, các bài toán tối ưu hàm submodular có thêm ràng buộc, xét với nhiễu, mở rộng không gian tìm kiếm... đôi khi sẽ làm thay đổi bản chất hàm mục tiêu, hoặc làm số lượng lời giải dự tuyển khó xác định. Do đó, cần các thuật toán mới giải quyết các vấn đề này.

Với bối cảnh nghiên cứu và các thách thức đặt ra ở trên, luận án định hướng nghiên cứu như sau:

- Nghiên cứu các bài toán tối đa hàm submodular SM, bài toán đối ngẫu SC với các ràng buộc kèm theo. Trong đó, ràng buộc chi phí được quan tâm, do tính tổng quát của nó. Từ đó, nghiên cứu các biến thể của hai bài toán trên: giải với

nhiễu, mở rộng hàm mục tiêu...;

- Nghiên cứu về thuật toán xấp xỉ và các phương pháp cải tiến hiện nay;
- Đề xuất các thuật toán xấp xỉ cạnh tranh giải quyết các bài toán.

Từ đó, luận án đã đề xuất các thuật toán xấp xỉ để giải quyết ba bài toán cụ thể: Tối đa hàm k-submodular với ràng buộc chi phí (kSMK), tối đa hàm submodular với ràng buộc chi phí có nhiễu (SMKN) và tối đa hàm Phủ Submodular trên lưới nguyên (DRSC).

Với bài toán kSMK, như đã phân tích, với các ứng dụng cần phân hoạch các phần tử vào k tập không giao nhau đòi hỏi sự mở rộng nghiên cứu tối ưu hàm submodular thành tối ưu hàm k-submodular. Các nghiên cứu hiện nay đối với bài toán kSMK còn nhiều hạn chế như chưa nhiều công bố có liên quan, vấn đề thời gian chạy và tỉ lệ xấp xỉ cần được cải thiện [134, 115]. Tỉ lệ xấp xỉ tốt nhất hiện nay đối với bài toán là $1/2$ của [140] nhưng lại không khả thi, không có tính thực tế, tốn lượng truy vấn quá lớn. Do đó, luận án đặt vấn đề đề xuất các thuật toán cho tỉ lệ xấp xỉ cạnh tranh với các nghiên cứu tiên tiến hiện nay nhưng giảm được độ phức tạp truy vấn.

Với bài toán SMKN, luận án đặt vấn đề đưa mô hình nhiễu khi xem xét lời giải cho bài toán SMK. Một số tác giả đã nghiên cứu nhiễu và ảnh hưởng của nó đối với bài toán tối ưu hàm submodular [40, 106], nhưng với ràng buộc khác. Với SMK chưa có nghiên cứu nào được đặt ra. Vấn đề thách thức đến từ ước lượng nhiễu F của hàm f, tính toán với nhiễu trong thuật toán, và ràng buộc chi phí làm cho bài toán càng có tính thách thức khi cần nhanh chóng lựa chọn được phần tử đưa vào tập lời giải. Thêm vào đó, sử dụng thuật toán luồng để cải thiện vấn đề thời gian chạy và bộ nhớ tiêu tốn so với tham lam đang là hướng nghiên cứu phổ biến hiện nay.

DRSC là một bài toán phù hợp với các kịch bản cần tối thiểu chi phí nhân lực, tài nguyên mà vẫn đạt năng suất cần thiết. Bài toán này còn có ý nghĩa thực tiễn khi một cách tự nhiên, chúng ta luôn lựa chọn nhiều lần những cá thể, đối tượng cho kết quả tốt trong công việc. Tuy nhiên, bài toán này đang tồn tại nhiều thách thức khi tính chất submodular được mở rộng. Thêm vào đó, thách thức rất lớn đến từ không gian tìm kiếm khổng lồ $\mathbb{Z}_+^V$ ảnh hưởng đến thời gian chạy. Trong khi đó, hướng nghiên cứu về bài toán này là nghiên cứu mới, có ít công bố liên quan. Một số công bố [52, 119] chỉ làm việc với hàm f là số nguyên, còn một số [129, 47] khác lại cho độ phức tạp truy vấn lớn. Do đó, nghiên cứu giải DRSC cho tỉ lê xấp xỉ cạnh tranh, sử dụng song song hóa với độ phức tạp song song và độ phức tạp truy vấn thấp được đặt ra.

1.6. Kết luận chương

Luận án đề cập đến bối cảnh nghiên cứu của bài toán tối ưu hàm submodular là một bài toán tối ưu tổ hợp quan trọng và phổ biến. Hàm submodular có tác dụng lớn trong các bài toán thu thập, xử lý thông tin và làm đa dạng dữ liệu.

Nhận thức được tiềm năng của bài toán tối đa hàm mục tiêu submodular, luận án đã chọn nghiên cứu đưa ra các thuật toán xấp xỉ hiệu quả cho bài toán tối đa hàm submodular có ràng buộc. Việc tìm lời giải cho các bài toán có nhiều thách thức. Hướng nghiên cứu chính là: (1) Giải bài toán tối đa hàm submodular có ràng buộc, trong đó, luận án tập trung vào ràng buộc chi phí và các biến thể của bài toán tối đa hàm submodular với ràng buộc. Các thuật toán xấp xỉ được thiết kế theo hướng là thuật toán tất định với tỉ lệ xấp xỉ là hằng số; (2) Các phương pháp cải tiến thuật toán xấp xỉ tham lam nhằm giảm thời gian chạy, bộ nhớ lưu trữ và số lượng truy vấn mà vẫn đảm bảo chất lượng lời giải.

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

Bài toán tối ưu hàm submodular là một trong các bài toán tối ưu tổ hợp thường gặp và có tính ứng dụng rộng rãi trong nhiều lĩnh vực như tối đa ảnh hưởng, hỗ trợ ra quyết định nhờ thu thập thông tin [81], phân đoạn ảnh [18, 78], tóm tắt tài liệu [98, 90] và học tích cực [60]...

Tuy nhiên, việc sử dụng mô hình tối ưu hàm submodular chưa đủ để đáp ứng cho một vài tình huống xảy ra khi cần tìm giải pháp để phân lớp các phần tử vào nhiều tập khác nhau nhằm thu thập thông tin với nhiều chủ đề khác nhau. Khi đó, một tập dữ liệu được phân hoạch thành k tập không giao nhau, mỗi một phần tử của tập dữ liệu cần lựa chọn đưa vào tập i nào đó trong k tập con sao cho lượng thông tin thu thập được có lợi nhất. Submodular được mở rộng thành hàm k-submodular. Các bài toán tối ưu hàm k-submodular đã trở thành một chủ đề nghiên cứu thu hút được nhiều sự quan tâm gần đây [135, 109, 106, 152].

Tiếp cận với xu hướng nghiên cứu này, luận án giải bài toán *tối đa hàm k-submodular với ràng buộc chi phí (k-Submodular Maximization under Knapsack constraint - kSMK)* bằng các thuật toán xấp xỉ. Bài toán kSMK có thể được vận dụng để giải quyết nhiều kịch bản trong thực tiễn như tối đa ảnh hưởng của k chủ đề, đặt k cảm biến, phân lớp... với giới hạn chi phí về nhân lực, ngân sách, thời gian... Tuy nhiên, các nghiên cứu cho kSMK còn chưa nhiều, bộc lộ một số hạn chế với vấn đề thời gian chạy và tỉ lệ xấp xỉ của thuật toán.

Trong chương này, luận án trình bày các thuật toán xấp xỉ tất định cho bài toán kSMK với hai trường hợp, hàm mục tiêu đơn điệu và không đơn điệu. Trong đó, các thuật toán cho hàm mục tiêu đơn điệu cho kết quả vượt trội với tỉ lệ xấp xỉ tốt hơn tỉ lệ có được của thuật toán xấp xỉ tốt nhất hiện nay mà độ phức tạp truy vấn giảm xuống tuyến tính. Các thuật toán cho hàm mục tiêu không đơn điệu cho kết quả tốt với tỉ lệ xấp xỉ tốt tương đương với tỉ lệ có được của thuật toán xấp xỉ tốt nhất hiện nay nhưng độ phức tạp truy vấn cũng giảm xuống tuyến tính.

2.1. Phát biểu bài toán, hàm mục tiêu và một số quy ước quan trọng

2.1.1. Phát biểu bài toán

Định nghĩa bài toán tối đa hàm k-submodular được xây dựng tương tự như bài toán tối đa hàm submodular với phát biểu như sau:

Định nghĩa 2.1 (Bài toán tối đa hàm k-submodular)**.** Cho một số nguyên dương $k > 0$, tập cơ sở V được phân hoạch thành k tập không giao nhau, $V = \{V_1, V_2, \ldots, V_k\}$ và hàm mục tiêu $f : (k+1)^V \mapsto \mathbb{R}_+$ là hàm k-submodular, bài toán cần tìm tập

lời giải $\mathbf{s} = \{S_1, S_2, \ldots, S_k\}$, $\forall S_i \subseteq V$, $1 \leq i \leq k$, thỏa mãn ràng buộc $\mathcal{C}$ đối với tập $\mathbf{s}$ sao cho $f(\mathbf{s})$ là cực đại:

$$\begin{aligned} &\max f(\mathbf{s}) \\ &\text{s.t } \mathbf{s} \in (k+1)^V. \end{aligned} \tag{2.1}$$

Khi bài toán yêu cầu giới hạn về ngân sách cho trước, $\mathcal{C}$ là ràng buộc chi phí. Ta có phát biểu về bài toán tối đa hàm k-submodular với ràng buộc chi phí tương ứng như sau:

Định nghĩa 2.2 (Bài toán kSMK)**.** Với ngân sách giới hạn $B > 0$ cho trước, mỗi phần tử $e \in V$ có một chi phí dương $c(e) \leq B$. Bài toán kSMK yêu cầu tìm một k-tập $\mathbf{s} = (S_1, S_2, \ldots, S_k)$ với tổng chi phí $c(\mathbf{s}) = \sum_{i \in [k]} \sum_{e \in S_i} c(e) \leq B$ sao cho $f(\mathbf{s})$ là cực đại.

Trong đó, V là tập cơ sở, k là một số nguyên dương, quy ước $[k] = \{1, 2, \ldots, k\}$ và $(k+1)^V = \{(V_1, V_2, \ldots, V_k) | V_i \subseteq V, \forall i \in [k], V_i \cap V_j = \emptyset, \forall i \neq j\}$ là họ k tập không giao nhau được gọi là *k-tập (k-set).*

2.1.2. Hàm mục tiêu và một số quy ước quan trọng

Để thuận tiện trong việc đưa ra các phát biểu tính chất hàm mục tiêu, xây dựng các thuật toán và phân tích lý thuyết, Bảng 2.1 tóm tắt các ký hiệu phổ biến để giải quyết bài toán kSMK.

Ký hiệu	Mô tả
V	Tập cơ sở $V = \{e_1, e_2, .., e_n\}$
n	Kích cỡ của tập V
$c(e)$	Chi phí của một đỉnh $e \in V$ bất kỳ
S	Tập các phần tử
X_i	Tập con thứ i của $\mathbf{x}$
$[k]$	Tập các số nguyên dương liên tiếp, $[k] = \{1, 2, \ldots, k\}$
$\mathbf{x}$	k-tập (k-set) trên $(k+1)^V$ gồm k tập con không giao nhau, $\mathbf{x} = \{X_1, X_2, \ldots, X_k\}$
$\mathbf{o}$, opt	$\mathbf{o}$ là vector lời giải tối ưu, opt $= f(\mathbf{o})$
$supp_i(\mathbf{x})$	$supp_i(\mathbf{x}) = X_i$
$supp(\mathbf{x})$	$supp(\mathbf{x}) = \cup_{i \in [k]} X_i$, là tập hợp mọi phần tử của k-tập $\mathbf{x}$, hay $\lvert\mathbf{x}\rvert = \lvert supp(\mathbf{x})\rvert$
(e, i)	Bộ (tuple) gồm phần tử $e \in V$ và trị trí i của nó trong k set
$\mathbf{x}(e)$	Vị trí của e trong k-tập $\mathbf{x}$, $\mathbf{x}(e) = i$
$\sqcup, \sqcap, \sqsubseteq$	Các phép toán trên $(k+1)^V$
B	Ngân sách cho trước, $B \in \mathbb{Z}_+$
$\Delta_{(e,i)} f(\mathbf{x})$	Lợi nhuận biên (marginal gain) khi thêm một bộ (e, i) vào $\mathbf{x}$

Bảng 2.1: Các ký hiệu thường dùng trong bài toán kSMK

Cụ thể, các định nghĩa và quy tắc đặt tên được sử dụng trong luận án như sau:

- Một tập $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ bất kỳ được gọi là là k-tập; Các tập con của một k-tập là không giao nhau; Một k-tập rỗng là $\mathbf{0} = (\emptyset, \ldots, \emptyset)$;

- Đặt $supp_i(\mathbf{x}) = X_i$, $supp(\mathbf{x}) = \cup_{i\in[k]} X_i$, với X_i là tập con thứ i của $\mathbf{x}$. Như vậy, ta có lực lượng của tập $\mathbf{x}$: $|\mathbf{x}| = |supp(\mathbf{x})|$;

- Cho $\mathbf{x} = (X_1, X_2, \ldots, X_k), \mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, ta thiết lập nếu $e \in X_i$ thì $\mathbf{x}(e) = i$ và i được gọi là vị trí của e, ngược lại thì $\mathbf{x}(e) = 0$. i là tập con nào đó trong k tập con của k-tập $\mathbf{x}$. Thêm một phần tử $e \notin supp(\mathbf{x})$ vào X_i có thể được biểu diễn thành $\mathbf{x} \sqcup (e, i)$. Một (e, i) được gọi là một bộ (tuple). Như vậy, khi thêm một phần tử e bất kỳ vào trong k-tập $\mathbf{x}$ được thể hiện bằng việc thêm bộ (e, i) vào $\mathbf{x}$.

- Để cho phân tích thuật toán, biểu diễn $\mathbf{x} = \{(e_1, i_1), (e_2, i_2), \ldots, (e_t, i_t)\}$ đối với $e_j \in supp(\mathbf{x}), i_j = \mathbf{x}(e_j), \forall 1 \leq j \leq t$.

- Khi $X_i = \{e\}$, và $X_j = \emptyset, \forall j \neq i$, $\mathbf{x}$ được trở thành (e, i).

- Có $\mathbf{x} \sqsubseteq \mathbf{y}$ khi và chỉ khi $X_i \subseteq Y_i \; \forall i \in [k]$.

Các quy ước trên làm cơ sở để các tính chất của hàm mục tiêu k-submodular được phát biểu. Các tính chất này đóng vai trò rất quan trọng trong việc phân tích các mối quan hệ giữa lời giải dự tuyển và lời giải tối ưu khi thiết kế các thuật toán. Ward và cộng sự [144] đã phát biểu các tính chất k-submodular như sau:

Định nghĩa 2.3 (Hàm k-submodular [144])**.** Hàm $f : (k+1)^V \mapsto \mathbb{R}_+$ là k-*submodular* khi và chỉ khi với các biến $\mathbf{x}, \mathbf{y}$ bất kì, $\mathbf{x} = (X_1, X_2, \ldots, X_k)$ và $\mathbf{y} = (Y_1, Y_2, \ldots, Y_k) \in (k+1)^V$, ta có:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}). \tag{2.2}$$

Trong đó:

$$\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, \ldots, X_k \cap Y_k)$$

và

$$\mathbf{x} \sqcup \mathbf{y} = (Z_1, \ldots, Z_k), \text{ với } Z_i = X_i \cup Y_i \setminus (\bigcup_{j\neq i} X_j \cup Y_j)$$

Ngoài ra, nếu f là một hàm đơn điệu tăng (monotone) thì hàm có thêm có tính chất sau:

Định nghĩa 2.4 (Hàm đơn điệu tăng [144])**.** Với bất kỳ $\mathbf{x} \in (k+1)^V, e \notin supp(\mathbf{x})$ và $i \in [k]$, ta có *lợi nhuận biên (marginal gain)* của một hàm k-submodular là

không âm, được cho bởi:

$$\begin{aligned}\Delta_{(e,i)}f(\mathbf{x}) =& f(X_1, \ldots, X_{i-1}, X_i \cup \{e\}, X_{i+1}, \ldots, X_k) \\ & - f(X_1, \ldots, X_k) \geq 0.\end{aligned}$$

Bên cạnh đó, nhằm chứng minh các hệ quả và bổ đề quan trọng của thuật toán, ta cần sử dụng các tính chất tương đương của hàm k-submodular. Một hàm f là k-submodular thỏa mãn 2 tính chất là *trực giao submodular (orthant submodular)* và *submodular từng cặp (pairwise submodular)*. Các tác giả [144] phát biểu một định lý quan trọng sau chỉ ra mối quan hệ giữa các tính chất này đối với hàm k-submodular:

Định lý 2.1 (Theo Định lý 7 của [144])**.** *Hàm f là k-submodular khi và chỉ khi nó thoả mãn tính chất trực giao submodular và submodular từng cặp.*

Hai tính chất trực giao submodular và submodular từng cặp được phát biểu như sau:

Định nghĩa 2.5 (Trực giao submodular)**.**

$$\Delta_{(e,i)}f(\mathbf{x}) \geq \Delta_{(e,i)}f(\mathbf{y})$$

Với bất kỳ $\mathbf{x}, \mathbf{y} \in (k+1)^V, e \notin supp(\mathbf{y}), \mathbf{x} \sqsubseteq \mathbf{y}$ và $i \in [k]$;

Định nghĩa 2.6 (Submodular từng cặp)**.** Với bất kỳ $i, j \in [k], i \neq j$:

$$\Delta_{(e,i)}f(\mathbf{x}) + \Delta_{(e,j)}f(\mathbf{x}) \geq 0.$$

Trong bài toán ta chỉ xét trường hợp $k \geq 2$, vì nếu $k = 1$ hàm k-submodular trở thành submodular. Với bối cảnh của bài toán kSMK, ta giả sử rằng các phần tử e có chi phí $c(e) > 0$ thỏa mãn $c(e) \leq B$, vì nếu không thỏa mãn điều kiện như vậy chỉ đơn giản là không xét phần tử đó là được. Ngoài ra, bài toán cũng giả định đã biết trước cách tính hàm f. Hay ta luôn giả sử tồn tại một hộp đen (blackbox) để với mọi tập $\mathbf{x}$, đều trả về giá trị của $f(\mathbf{x})$. Khi đó, ta nói bài toán luôn giả sử đã cho trước một ước lượng truy vấn (oracle query) của hàm $f(\cdot)$. Đồng thời, giả sử hàm f được chuẩn hóa, tức là $f(\mathbf{0}) = 0$, mang hàm ý tập hợp rỗng không mang lại giá trị đóng góp về mặt lợi ích.

2.2. Ứng dụng của bài toán

Tối đa hàm k-submodular giải quyết nhiều kịch bản mà tối đa hàm submodular chưa thể giải quyết được. Có thể chỉ ra một vài ví dụ như các trường hợp bên

dưới. Khi các ứng dụng này bị ràng buộc bởi thời gian, nhân lực hay ngân sách, các ứng dụng trở thành các thể hiện cụ thể của kSMK.

1. **Bài toán tối đa ảnh hưởng theo k chủ đề.** Ví dụ chiến dịch quảng cáo cần lan truyền tiếp thị k sản phẩm hoặc k chủ đề khác nhau tới tập người dùng sao cho ảnh hưởng đến nhiều người dùng nhất.

Bản chất toán học của bài toán là tối đa hóa hàm k-submodular theo mô hình khuếch tán. Ban đầu, mô hình này do Kempe và cộng sự [43] đề xuất để mô tả cơ chế khuếch tán của một loại ảnh hưởng duy nhất của một tập hạt giống tới toàn bộ tập cơ sở. Các tác giả trong [109] đã khái quát hóa mô hình này để cho phép $k \geq 2$ các loại ảnh hưởng tham gia lan truyền thông tin. Bài toán được xây dựng thành *tối đa ảnh hưởng theo k chủ đề (k-topic Influence Maximization kIM)*. Bài toán này thường xuyên được sử dụng trong các nghiên cứu về k-submodular [151, 106, 115].

2. **Bài toán tối đa hóa độ phủ của thông tin theo k chủ đề.** Bài toán tối đa ảnh hưởng ở trên chưa xét đến tình huống một phần tử không bị ảnh hưởng nhưng nó vẫn được thông báo về chủ đề đang lan truyền từ ít nhất một hàng xóm đã bị ảnh hưởng của nó. Wang và cộng sự [143] đã nghiên cứu bài toán *tối đa hóa phạm vi phủ thông tin* nhằm tối đa hóa theo kỳ vọng bao gồm các phần tử được kích hoạt và cả các phần tử được thông báo về chủ đề đang được lan truyền. Sau đó, Qian và cộng sự [116] đã phát triển thành bài toán *tối đa hóa phạm vi phủ thông tin của k chủ đề (k-topic information Coverage Maximization - kCM)*.

3. **Bài toán đặt k loại cảm biến.** Bài toán mở rộng từ bài toán đặt cảm biến sao cho tối đa thông tin thu được. Giả sử ta có k loại cảm biến về nhiệt độ, tốc độ gió và năng lượng,... và một tập hợp V gồm n vị trí để đặt chúng. Sơ đồ vị trí k loại cảm biến được thể hiện bằng mỗi k-bộ (còn gọi là k-tuple) của các vị trí trong V và loại cảm biến được đặt nếu mỗi vị trí chỉ được phép có một cảm biến. Bài toán trở thành đặt cảm biến i thuộc k loại cảm biến đã cho ở vị trí nào để tối đa lượng thông tin thu được. Lúc này, bài toán được gọi là *đặt k loại cảm biến (k-type Sensor Placement - kSP)*.

4. **Một số ứng dụng khác:** Ngoài các ứng dụng nói trên, còn nhiều ứng dụng khác cũng được xây dựng dựa trên bài toán tối đa hàm k-submodular như:

- *Max-k cut*. Đây là một bài toán nổi tiếng minh họa rằng cực đại một hàm submodular là bài toán NP-khó ngay cả khi hàm mục tiêu f đã được cho một cách tường minh. Ward và các cộng cự [144] đã chỉ ra rằng bài toán Max-k cut có hàm mục tiêu là k-submodular.

- *Lựa chọn đặc trưng cho k lớp*. Lựa chọn đặc trưng tăng cường phân tích các tập dữ liệu lớn bằng cách giảm kích thước của dữ liệu. Kết quả là các bài toán lựa chọn nhiều đặc trưng bao gồm nhóm các tính năng liên quan đến k lớp và các biến

dự đoán không tương quan với k, từ đó hình thành bài toán *lựa chọn đặc trưng cho k lớp (k-class feature selection)*. Bài toán không chỉ yêu cầu tìm các đặc trưng nhiều thông tin nhất mà còn yêu cầu phân loại các đặc trưng liên quan đến các biến dự đoán, dẫn đến vấn đề tối đa hóa k-submodular [125, 121].

2.3. Các thách thức của bài toán

Có thể thấy các ứng dụng của kSMK rất nhiều trong thực tiễn. Tuy nhiên, để giải quyết tốt bài toán này cần vượt qua một số thách thức của hàm mục tiêu và vấn đề thời gian chạy.

- Thứ nhất, vì tối đa hóa hàm submodular là bài toán NP-khó, nên tối đa hóa hàm k-submodular nói chung và bài toán kSMK nói riêng cũng là một bài toán NP-khó. Vì vậy cần các thuật toán xấp xỉ hiệu quả để giải bài toán này;

- Thứ hai, không gian tìm kiếm lời giải tăng theo hàm mũ khi kích cỡ n của tập dữ liệu tăng lên. Vì vậy, việc đề xuất các thuật toán xấp xỉ đảm bảo chất lượng lời giải trong thời gian đa thức là một thách thức cốt lõi cần phải vượt qua;

- Thứ ba, thách thức đến từ tính chất k-submodular của hàm mục tiêu. Hàm k-submodular và hàm submodular khác nhau về bản chất. Do vậy, áp dụng các phương pháp phân tích lý thuyết của submodular trên k-submodular chưa chắc đã đạt hiệu quả tương tự, hoặc cần các biến đổi phức tạp hơn. Bài toán tối đa hàm k-submodular sẽ là không tầm thường vì sự mở rộng của k-submodular so với submodular khiến các phân tích và thuật toán tốt nhằm tối đa hàm submodular chưa chắc đã phát huy tác dụng với k-submodular. Các thuật toán xấp xỉ cho chất lượng lời giải tốt do đó cũng bị ảnh hưởng. Vì vậy, tối đa hàm k-submodular cần các đóng góp về thuật toán hiệu quả đảm bảo cân bằng giữa tỉ lệ xấp xỉ và thời gian chạy và độ phức tạp truy vấn.

- Thứ tư, thách thức từ ràng buộc chi phí. Khác với ràng buộc về lực lượng hay matroid chỉ cần liệt kê các phần tử không vượt quá kích cỡ tập lời giải cho trước, ràng buộc chi phí làm cho có thể có nhiều tập lời giải với các kích cỡ khác nhau. Do đó, việc lựa chọn các lời giải cũng phức tạp hơn so với các ràng buộc còn lại.

- Thứ năm, các nghiên cứu về kSMK hiện nay còn chưa nhiều. Như đã khái quát ở trên, chủ yếu các nghiên cứu đang tập trung với ràng buộc lực lượng. Các nghiên cứu đã có với ràng buộc lực lượng không thể áp dụng trực tiếp lên ràng buộc chi phí.

- Thứ sáu, yêu cầu cần giải quyết hàm mục tiêu không còn đơn điệu để bài toán kSMK phù hợp với nhiều tình huống trong thực tiễn hơn.

- Cuối cùng, xu thế nghiên cứu hiện nay về tối ưu hàm submodular là giảm độ phức tạp truy vấn xuống còn tuyến tính. Việc nghiên cứu đối với bài toán kSMK

cũng không nằm ngoài xu thế này. Cũng tương tự như các bài toán tối đa hàm submodular, bài toán tối đa hàm k-submodular cần truy xuất hàm mục tiêu nhiều lần khi tìm phần tử tốt để thêm vào tập lời giải. Các thuật toán tham lam có độ phức tạp truy vấn lớn, ảnh hưởng tới thời gian chạy của thuật toán. Do vậy cần đề xuất các thuật toán xấp xỉ hiệu quả cho lời giải cạnh tranh, đồng thời giảm độ phức tạp truy vấn.

Như vậy, để giải bài toán tối đa hàm k-submodular với ràng buộc chi phí bằng các thuật toán cải tiến hiệu quả, cần khảo sát, phân tích và vận dụng nhiều đóng góp khác nhau của giới học thuật trên thế giới về tối ưu submodular và tối ưu k-submodular. Phần tiếp theo sẽ tóm tắt nội dung một số nghiên cứu có liên quan.

2.4. Các vấn đề nghiên cứu có liên quan

Ban đầu, các tác giả nghiên cứu bài toán tối đa hàm k-submodular không có ràng buộc bằng nhiều phương pháp khác nhau như: thiết kế thuật toán tham lam tất định [144], tham lam ngẫu nhiên [73], giải ngẫu nhiên [110], và thiết kế thuật toán luồng [126]... Tuy nhiên sau này, các nghiên cứu chuyển sang giải với ràng buộc nhiều hơn do có sự gần gũi với các bài toán xảy ra trong thực tế.

Đầu tiên, các nghiên cứu tập trung vào ràng buộc lực lượng[109, 118, 152]. Tuy nhiên, ràng buộc chi phí là ràng buộc tổng quát hơn, khi xét mỗi một phần tử $e \in V$ sẽ cần ít nhất một chi phí $c(e)$ để hoạt động. Như đã phân tích ở chương trước, khi $c(e) = 1$, bài toán với ràng buộc chi phí trở về với ràng buộc lực lượng. Thậm chí, bài toán tối đa hàm k-submodular với ràng buộc chi phí còn nâng độ khó của bài toán lên nếu với mỗi phần tử e có chi phí $c_i(e)$ riêng biệt cho từng $i \in [k]$ tập con tương ứng. Như vậy, giải bài toán lúc này sẽ gặp phải nhiều thách thức hơn do xuất hiện rất nhiều lời giải với chi phí khác nhau.

Các công bố đầu tiên giải với tối đa hàm k-submodular đơn điệu. Tuy nhiên, nghiên cứu hàm mục tiêu không đơn điệu góp phần làm tổng quát hoá bài toán vì đóng góp lợi ích của một phần tử vào một tập hợp không phải lúc nào cũng tăng mà có thể giảm. Điều này đã được chỉ ra trong các nghiên cứu của [137, 97] về tối ưu hàm submodular. Hệ quả là hàm mục tiêu k-submodular cũng có thể không đơn điệu. Vì thế việc nghiên cứu hàm k-submodular không đơn điệu cần những đóng góp mới có giá trị.

Bài toán kSMK lần đầu tiên được quan tâm bởi Tang và cộng sự [134], trong đó các tác giả đã đề xuất một thuật toán tham lam xấp xỉ với tỉ lệ $1/2 - 1/(2e)$. Thuật toán của họ là sự kế thừa từ thuật toán tham lam của Sviridenko [131]. Tuy nhiên, thuật toán tạo ra độ phức tạp truy vấn khá lớn là $O(n^4k^3)$ khiến nó khó áp dụng đối với các bài toán có kích thước dữ liệu đầu vào cỡ lớn, cho dù khi thuật

toán đó có thể tính toán hàm mục tiêu f trong thời gian là $O(1)$.

Bên cạnh đó, các tác giả trong [141] đề xuất một phương pháp khác, là mở rộng hàm mục tiêu sang liên tục, còn gọi là phương pháp *mở rộng đa tuyến tính (multi-linear extension)* cho kSMK. Thuật toán mà họ đưa ra dựa trên tính toán ngẫu nhiên cho tỉ lệ xấp xỉ lên tới $1/2 - 2\epsilon$ theo kỳ vọng. Công trình của họ cho tỉ lệ xấp xỉ tốt nhất, tuy nhiên thuật toán này không thực tế do độ phức tạp truy vấn lớn của phần mở rộng liên tục.

Một trong các phương pháp hiệu quả góp phần làm giảm dung lượng lưu trữ và giảm thời gian tính toán khi dữ liệu đầu vào có kích thước lớn là thiết kế *thuật toán luồng*. Thiết kế thuật toán luồng trở thành một phương pháp được ưa chuộng để giải các bài toán tối đa hàm submodular với nhiều loại ràng buộc khác nhau như ràng buộc về lực lượng [62, 5, 89, 85], chi phí [70], k tập [66] và matroid [26]... Tuy nhiên, các phương pháp hiện có không dễ dàng khi áp dụng trực tiếp vào bài toán kSMK vì sự khác nhau về mặt bản chất giữa tính submodular và k-submodular.

Đầu tiên, áp dụng luồng để tối đa hàm k-submodular đã được đề cập trong [144, 73]. Các tác giả đã đề xuất thuật toán gọi là *thuật toán luồng quét 1 lần (single-pass streaming)*. Thuật toán này chỉ duyệt từng phần tử của tập cơ sở 1 lần. Lưu ý rằng việc chọn ra các phần tử có đóng góp lợi ích lớn nhất để bổ sung vào tập lời giải trong thuật toán tham lam, các thuật toán phải duyệt tuần từ tập cơ sở V nhiều hơn 1 lần. Ý tưởng của các tác giả trong [73] là khai thác lựa chọn ngẫu nhiên trong khi các tác giả trong [144] lại phát triển thuật toán dựa trên một phân phối mới khi lựa chọn một phần tử với các chi phí khác nhau, sau đó thiết lập một mối quan hệ giữa lời giải đang xét với lời giải tối ưu.

Gần đây một nhóm nghiên cứu đến từ Việt Nam, Pham và cộng sự [115] đã đề xuất 2 thuật toán luồng 1 lần quét cho bài toán kSMK với độ phức tạp truy vấn là $O(nk\log(n)/\epsilon)$. Bài toán kSMK là trường hợp đặc biệt trong nghiên cứu của họ khi $\beta = 1$. Khi đó, các thuật toán của họ đạt tỉ lệ xấp xỉ là $1/4-\epsilon$ và $k/(4k-1)-\epsilon$ theo kỳ vọng cho trường hợp f đơn điệu, và $1/5 - \epsilon$ và $k/(5k-3) - \epsilon$ theo kỳ vọng khi f không đơn điệu. Các tác giả này đã lần đầu tiên đề xuất hai thuật toán luồng cho kSMK để đạt được tỉ lệ xấp xỉ là hằng số trong khi các truy vấn chỉ còn là $O(kn\log(n))$. Tỉ lệ xấp xỉ là hằng số giúp cho chất lượng lời giải không phụ thuộc vào ngân sách B và kích thước của tập dữ liệu đầu vào. Điều này thực sự có lợi khi kích thước tập dữ liệu đầu vào tăng lên.

Từ bối cảnh nghiên cứu trên, cho thấy để giải quyết bài toán kSMK cần tập trung vào đề xuất các thuật toán xấp xỉ tất định với tỉ lệ xấp xỉ là hằng số sao cho giảm độ phức tạp truy vấn cạnh tranh được với độ phức tạp tốt nhất hiện nay đang

là $O(kn\log(n))$ nhưng vẫn đảm bảo được chất lượng lời giải. Bảng 2.2 tóm tắt các công bố tốt nhất hiện nay đối với bài toán kSMK. Greedy là tên cho thuật toán tham lam đề xuất bởi Tang trong [134], DS và RS là tên viết tắt cho hai thuật toán luồng tất định và luồng ngẫu nhiên trong [115]. Độ phức tạp truy vấn $poly(n)$ là độ phức tạp đa thức.

Thuật toán	**Tỉ lệ xấp xỉ**	**Độ phức tạp truy vấn**	**Loại**
Greedy[134]	$1/2 - 1/(2e)$	$O(n^4k^3)$	Tất định
Thuật toán 3 trong [141]	$1/2$	$poly(n)$	Tất định
Thuật toán luồng tất định - DS [115]	$1/4 - \epsilon$	$O(kn\log(n)/\epsilon)$	Tất định
Thuật toán luồng ngẫu nhiên - RS [115]	$k/(4k-1) - \epsilon$	$O(kn\log(n)/\epsilon)$	Ngẫu nhiên

Bảng 2.2: Các thuật toán hiện nay cho kSMK.

2.5. Các thuật toán xấp xỉ cho bài toán kSMK đơn điệu tăng

2.5.1. Kết quả mới của luận án

Để giải quyết bài toán kSMK, luận án đã đề xuất một vài thuật toán cho lời giải với tỉ lệ xấp xỉ tốt tương đương hoặc thậm chí tốt hơn các thuật toán hiện có, đồng thời giảm được số lượng truy vấn hàm f. Số lượng truy vấn được thể hiện thông qua độ phức tạp truy vấn. Đây là một hướng nghiên cứu quan trọng, thể hiện sự cạnh tranh giữa các nghiên cứu hiện nay, vì độ phức tạp truy vấn ảnh hưởng trực tiếp tới thời gian chạy của thuật toán. Nhìn chung, nghiên cứu về bài toán kSMK, luận án tập trung vào:

1. Đề xuất các thuật toán tất định cho tỉ lệ xấp xỉ hằng số cạnh tranh;

2. Các thuật toán giảm được độ phức tạp truy vấn xuống còn tuyến tính;

3. Nghiên cứu giải bài toán kSMK với hai trường hợp hàm mục tiêu đơn điệu và không đơn điệu.

4. Tiến hành thực nghiệm so sánh các thuật toán được đề xuất và các thuật toán tốt nhất hiện nay.

Đối với trường hợp hàm mục tiêu đơn điệu, luận án trình bày về các thuật toán mới cho bài toán kSMK, đáp ứng một số yêu cầu về đảm bảo hiệu quả và giảm độ phức tạp của truy vấn. Các kết quả nghiên cứu là đầu tiên cung cấp tỉ lệ xấp xỉ không đổi chỉ trong phạm vi độ phức tạp truy vấn là $O(kn)$ và có thể trả về tỉ lệ xấp xỉ $1/3 - \epsilon$.

Với kết quả nêu trên, các đóng góp của luận án là tốt hơn thuật toán tốt nhất hiện nay của Tang và cộng sự [134] khi giải bài toán này bằng thuật toán thời gian đa thức tất định, thiết kế dựa trên tham lam với tỉ lệ xấp xỉ là $1/2 - 1/(2e)$. Đóng góp của luận án là cho chi phí tính toán thấp nhất so với bất kỳ thuật toán xấp xỉ

có tỉ lệ không đổi nào và đóng một vai trò quan trọng trong việc tìm kiếm các giải pháp gần tối ưu cho các ứng dụng. Điều này có giá trị không nhỏ vì chi phí đánh giá hàm mục tiêu f có thể rất tốn kém.

Tổng thể, những đóng góp của luận án đối với bài toán kSMK đơn điệu tăng bao gồm:

- Đề xuất thuật toán xấp xỉ nhanh, **FA** (Fast Approximation, Thuật toán 2), cho tỉ lệ xấp xỉ là $1/10$, cần 1 lần quét tập cơ sở với độ phức tạp truy vấn là $O(kn)$. Thuật toán được xây dựng dựa trên ý tưởng chia đôi tập cơ sở V thành 2 tập nhằm tìm giải pháp tối ưu trên tập thứ nhất và tìm giải pháp gần tối ưu trên tập thứ hai. Đây là thuật toán đơn giản đầu tiên nhưng quan trọng bởi vì nó giới hạn khoảng của giá trị tối ưu và cung cấp một chiến thuật chia tập dữ liệu để giảm độ phức tạp truy vấn còn $O(nk)$.

- Đề xuất thuật toán xấp xỉ nhanh cải tiến, **IFA** (Improved Fast Approximation, Thuật toán 3), cho tỉ lệ xấp xỉ là $1/4 - \epsilon$, yêu cầu độ phức tạp truy vấn là $O(kn/\epsilon)$ với $\epsilon \in (0, 1)$ là tham số chính xác. Thuật toán được xây dựng dựa trên ngưỡng mật độ (density gain threshold) của các phần tử. Thuật toán này cho tỉ lệ xấp xỉ **tốt tương đương** thuật toán DS nêu trong [115], là một thuật toán tất định tốt nhất hiện nay.

- Chất lượng của **IFA** được cải tiến bằng thuật toán xấp xỉ cải tiến tăng cường, **IFA**+ (Improved Fast Approximation Plus, Thuật toán 4). Thuật toán này cần $O(kn \log(1/\epsilon)/\epsilon)$ truy vấn nhưng có thể cung cấp tỉ lệ xấp xỉ lên tới $(1/3 - \epsilon)$. Thuật toán được xây dựng dựa trên ngưỡng mật độ và tăng cường chất lượng của lời giải dự tuyển bằng cách phân hoạch lại lời giải dự tuyển và bổ sung thêm các phần tử tốt còn trong tập cơ sở. Đây là thuật toán cho **tỉ lệ xấp xỉ tốt hơn** so với thuật toán cho tỉ lệ xấp xỉ tốt nhất hiện nay là thuật toán Greedy đề xuất bởi Tang và cộng sự [134].

- Để kiểm nghiệm các đóng góp về mặt lý thuyết, luận án thực hiện một số thực nghiệm tổng quát với ba ứng dụng của kSMK, bao gồm: Tối đa ảnh hưởng theo k chủ đề với ràng buộc chi phí - kIMK, tối đa hóa mức độ bao phủ thông tin k chủ đề - kCMK và đặt vị trí k loại cảm biến - kSPK. Kết quả thử nghiệm đã chỉ ra rằng các thuật toán nêu trong luận án không chỉ yêu cầu số lượng truy vấn thấp hơn lên tới hàng trăm lần so với Greedy, thấp hơn tới 15-20 lần so với các thuật toán luồng (được đề cập trong Bảng 2.3) mà còn cho chất lượng lời giải tốt hơn khoảng 1.5 lần so các thuật toán luồng tốt nhất hiện nay.

Ngoài ra, các thuật toán nêu trong luận án có ưu thế vượt trội hơn các thuật toán còn lại khi B tăng và n tăng. Giá trị đóng góp này cho thấy các thuật toán của luận án có thể làm việc được với các tập dữ liệu có kích thước lớn.

Bảng 2.3 là sự mở rộng của Bảng 2.2 khi so sánh các thuật toán được đề xuất bởi luận án với các thuật toán tốt nhất hiện nay cho kSMK theo 3 khía cạnh: tỉ lệ xấp xỉ, độ phức tạp truy vấn, và thuật toán này có phải là tất định hay không (các ký hiệu của Bảng 2.2 giữ nguyên). Các trường này cho thấy rằng thuật toán của tác giả đề xuất đảm bảo số lượng truy vấn thấp hơn và tỉ lệ xấp xỉ xác định có giá trị tương đương hoặc thậm chí tốt hơn các thuật toán khác.

Thuật toán 3 của Wang và cộng sự [141] cho tỉ lệ xấp xỉ tốt nhất, nhưng độ phức tạp truy vấn lớn tùy ý, do phương pháp của các tác giả dựa trên biến đổi hàm mục tiêu từ hàm rời rạc thành hàm liên tục. Tiếp theo, thuật toán tham lam, Greedy, của Tang và cộng sự [134] cho tỉ lệ xấp xỉ tốt chỉ sau công bố của Wang, nhưng độ phức tạp truy vấn vẫn lớn, $O(n^4k^3)$. Thuật toán **IFA**+ trình bày trong luận án cho tỉ lệ xấp xỉ là $1/3 - \epsilon$, cho kết quả tốt hơn công bố của Tang mà độ phức tạp truy vấn còn là tuyến tính. Thuật toán **IFA** của luận án cho kết quả tốt tương đương thuật toán DS của [115] nhưng độ phức tạp truy vấn cũng thấp hơn. Tất cả các thuật toán nêu trong luận án đều có tỉ lệ xấp xỉ là hằng số và độ phức tạp truy vấn là tuyến tính.

Thuật toán	**Tỉ lệ xấp xỉ**	**Độ phức tạp truy vấn**	**Loại**
Greedy[134]	$1/2 - 1/(2e)$	$O(n^4k^3)$	Tất định
Thuật toán 3 trong [141]	$1/2$	$poly(n)$	Tất định
Thuật toán luồng tất định - DS [115]	$1/4 - \epsilon$	$O(kn\log(n)/\epsilon)$	Tất định
Thuật toán luồng ngẫu nhiên - RS [115]	$k/(4k-1) - \epsilon$	$O(kn\log(n)/\epsilon)$	Ngẫu nhiên
FA (Thuật toán 2 trong luận án)	$1/10$	$O(kn)$	Tất định
IFA (Thuật toán 3 trong luận án)	$1/4 - \epsilon$	$O(kn/\epsilon)$	Tất định
IFA+ (Thuật toán 4 trong luận án)	$1/3 - \epsilon$	$O(kn\log(1/\epsilon)/\epsilon)$	Tất định

Bảng 2.3: So sánh các thuật toán của luận án với các thuật toán hiện có cho kSMK.

2.5.2. Thuật toán xấp xỉ nhanh: FA

Phần này giới thiệu về thuật toán đầu tiên, thuật toán xấp xỉ nhanh **FA** (*Fast Approximation*). Ý tưởng chính của **FA** đó là:

- Chia tập cơ sở V thành 2 tập con V_1 và V_2. Tập con đầu tiên V_1 chứa các phần tử có chi phí lớn hơn $B/2$ và tập con thứ 2 V_2 chứa các phần tử còn lại. Thuật toán tìm bộ (e_m, i_m) cho $f(\cdot)$ lớn nhất. (e_m, i_m) nếu có trên tập V_1, nó là giải pháp tốt nhất hiện có (dòng 4 thuật toán). Vì các phần tử trên V_1 có chi phí lớn hơn $B/2$, nên trên tập này tìm được tối đa 1 bộ (e_m, i_m). Khi đó, (e_m, i_m) là lời giải tối ưu trên V_1. Tập con thứ hai sử dụng tập lời giải dự tuyển s và ngưỡng tần suất để đưa số lượng truy vấn về tuyến tính.
- Tìm và kết hợp các lời giải từ 2 tập con nêu trên để lấy về lời giải cho giá trị

hàm mục tiêu lớn nhất.

Cụ thể, thuật toán sẽ nhận đầu vào là một bộ, gọi là một thể hiện (V, f, k, B) của kSMK; và một lời giải dự tuyển $\mathbf{s}$ ban đầu là tập rỗng $\mathbf{0}$ và bộ (e_m, i_m) khởi tạo là $(\emptyset, 1)$. Bộ (e_m, i_m) dùng để cập nhật lời giải tối ưu thuộc tập con đầu tiên, còn lời giải dự tuyển $\mathbf{s}$ là để tìm lời giải gần tối ưu ở tập con thứ hai.

Ta sẽ đánh giá các phần tử e của tập V, với một phần tử e đang xét, thuật toán sẽ tìm vị trí tốt nhất của e trong k tập con, gọi là i_e theo nghĩa là tập con i trong k tập sao cho giá trị $f((e, i_e))$ là lớn nhất. Nếu chi phí của nó lớn hơn $B/2$ thì thuật toán sẽ cập nhật (e_m, i_m) là giải pháp tốt nhất của tập con đầu tiên hiện tại (dòng 4).

Mặt khác, thuật toán sẽ tìm vị trí $i'_e = \arg\max_{i \in [k]} \Delta_{(e,i_e)} f(\mathbf{s})$ (dòng 6, Thuật toán 2), và thêm bộ (e, i'_e) vào tập lời giải $\mathbf{s}$ nếu điều kiện $\Delta_{(e,i'_e)} f(\mathbf{s}) \geq c(e) f(\mathbf{s})/B$ được thỏa mãn. Sau khi vòng lặp chính kết thúc, thuật toán chọn k-tập $\mathbf{s}'$ là tập j bộ cuối cùng được thêm vào $\mathbf{s}$ với tổng chi phí lớn nhất gần B nhất (dòng 12). Cuối cùng, thuật toán trả về lời giải $\mathbf{s}_{final}$ là giải pháp tốt nhất giữa (e_m, i_m) và $\mathbf{s}'$. Chi tiết về thuật toán được trình bày đầy đủ trong Thuật toán 2.

Phân tích sâu hơn, thuật toán này giống với chiến lược "chia để trị", trong đó sử dụng một phép phân chia các tập con một cách thích hợp dựa trên chi phí của các phần tử. Việc chia tập cơ sở mang lại hiệu quả khi đồng thời thực hiện: (1) Tìm giải pháp tối ưu trên tập con đầu tiên trong thời gian tuyến tính. Vì ở trong tập này, lời giải khả thi chỉ có thể có nhiều nhất một phần tử; (2) Thuật toán tìm thấy giải pháp gần tối ưu trên tập thứ hai cũng trong thời gian tuyến tính.

Đối với việc tìm kiếm lời giải trong tập thứ 2, thuật toán đã được phát triển dựa trên ý tưởng của Kuhnle và cộng sự trong [85]. Trong bài báo này, các tác giả đã đánh giá các phần tử và lựa chọn phần tử nào có *mức tăng mật độ (density gain)*. Mức tăng mật độ là là *tỉ lệ giữa đóng góp lợi ích của phần tử vào lời giải hiện tại và chi phí của phần tử đó*. Tỉ lệ này phải không nhỏ hơn tỉ lệ giữa giá trị mục tiêu của tập lời giải hiện tại với ngân sách B thì phần tử đó sẽ được giữ lại và các phần tử khác thì bị loại bỏ. Ý tưởng của họ rất hiệu quả trong việc giảm số lượng của các truy vấn của thuật toán xấp xỉ với hệ số là hằng số. Tuy nhiên, để xử lý chi phí và k-submodular, chúng ta cần thực hiện một vài phân tích không tầm thường để đưa ra một tỉ lệ xấp xỉ.

Cuối cùng, thuật toán lấy t bộ cuối cùng được thêm vào lời giải dự tuyển $\mathbf{s}$ vì quy tắc chọn phần tử của thuật toán (dòng 7) chọn các phần tử cho lợi nhuận biên tăng quá ngưỡng $c(e)f(\mathbf{s})/B$ làm cho các phần tử chọn sau sẽ tốt hơn phần tử chọn trước đó.

Để phân tích tỉ lệ xấp xỉ, ta cần xây dựng mối quan hệ giữa lời giải cuối cùng

Algorithm 2 FA

Input: V, f, k, $B > 0$

Output: a solution $\mathbf{s}$

1: $\mathbf{s} \leftarrow \mathbf{0}$; $(e_m, i_m) \leftarrow (\emptyset, 1)$
2: **for all** $e \in V$ **do**
3: $\quad i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
4: $\quad (e_m, i_m) \leftarrow \arg\max_{(e', i') \in \{(e_m, i_m), (e, i_e)\}} f((e', i'))$
5: $\quad$ **if** $c(e) \leq B/2$ **then**
6: $\quad\quad i'_e \leftarrow \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s})$
7: $\quad\quad$ **if** $\Delta_{(e,i'_e)} f(\mathbf{s}) \geq c(e) f(\mathbf{s})/B$ **then**
8: $\quad\quad\quad \mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i'_e)$
9: $\quad\quad$ **end if**
10: $\quad$ **end if**
11: **end for**
12: $\mathbf{s}' \leftarrow \arg\max_{\mathbf{s}_j : j \leq t, c(\mathbf{s}_j) \leq B} c(\mathbf{s}_j)$, $t = |supp(\mathbf{s})|$, $\mathbf{s}_j = \{(e_{t-j+1}, i_{t-j+1}), (e_{t-j+2}, i_{t-j+2}), \ldots, (e_t, i_t)\}$ // last j tuples added into $\mathbf{s}$
13: $\mathbf{s}_{final} \leftarrow \arg\max_{\mathbf{s} \in \{(e_m, i_m), \mathbf{s}'\}} f(\mathbf{s})$
14: **return** $\mathbf{s}_{final}$

$\mathbf{s}_{final}$ với các lời giải (e_m, i_m) và $\mathbf{s}'$, giữa $\mathbf{s}'$ với $\mathbf{s}_j$ và lời giải tối ưu $\mathbf{o}_2$ trên tập V_2. Bên cạnh các quy ước chung đã được nêu trong Bảng 2.1, thuật toán này cần sử dụng bộ ký hiệu riêng dùng cho thuật toán và phân tích thuật toán. Các ký hiệu được định nghĩa như sau:

- $V_1 = \{e \in V : c(e) > B/2\}, V_2 = \{e \in V : c(e) \leq B/2\}$ là 2 tập con được chia theo chi phí của các phần tử từ tập V.

- $\mathbf{o}'_1 = \{(e, \mathbf{o}(e)) : e \in V_1\}, \mathbf{o}'_2 = \{(e, \mathbf{o}(e)) : e \in V_2\}$.

- $\mathbf{o}_1$ là lời giải tối ưu trên tập V_1.

- $\mathbf{o}_2$ là lời giải tối ưu trên tập V_2.

- (e_j, i_j) là phần tử thứ j được thêm vào vòng lặp chính của thuật toán.

- $\mathbf{s} = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: k-tập $\mathbf{s}$ gồm t bộ sau khi kết thúc vòng lặp chính, $t = |supp(\mathbf{s})|$.

- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: là k-tập $\mathbf{s}$ (trong vòng lặp chính) sau khi thêm j phần tử, $1 \leq j \leq t$. Như vậy, có thể viết $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.

- $\mathbf{s}_j = \{(e_{t-j+1}, i_{t-j+1}), (e_{t-j+2}, i_{t-j+2}), \ldots, (e_t, i_t)\}$ là j phần tử cuối cùng được thêm vào $\mathbf{s}$.

- $\mathbf{o}_2^j = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^j$. $\mathbf{o}_2^j$, bao gồm tất cả các phần tử của $\mathbf{o}_2$ có mặt trong $\mathbf{s}^j$ và các phần tử chỉ nằm trong $\mathbf{s}^j$.

- $\mathbf{o}_2^{j-1/2} = (\mathbf{o}_2 \sqcup \mathbf{s}^j) \sqcup \mathbf{s}^{j-1}$, lấy tất cả các phần tử trong $\mathbf{o}$ đã xuất hiện trong $\mathbf{o}^j$

nhưng chưa xuất hiện trong $\mathbf{o}^{j-1}$.

- $\mathbf{s}^{j-1/2}$: Nếu $e_j \in supp(\mathbf{o}_2)$, thì có $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1} \sqcup (e_j, \mathbf{o}_2(e_j))$. Nếu $e_j \notin supp(\mathbf{o}_2)$, $\mathbf{s}^{j-1/2} = \mathbf{s}^{j-1}$.

- $\mathbf{u}^t = \{(u_1, i_1), (u_2, i_2), \ldots, (u_r, i_r)\}$ là tập các phần tử ở trong tập $\mathbf{o}_2^t$ mà không ở trong tập $\mathbf{s}^t$, $r = |supp(\mathbf{u}^t)|$.

- $\mathbf{u}_l^t = \mathbf{s}^t \sqcup \{(u_1, i_1), (u_2, i_2), \ldots, (u_l, i_l)\}, \forall 1 \leq l \leq r$ và $\mathbf{u}_0^t = \mathbf{s}^t$.

Giả sử lời giải tối ưu có các phần tử: $\mathbf{o} = \{(o_1, i_1^*), \ldots, (o_m, i_m^*)\}$, đặt $m = |supp(\mathbf{o})|$. Không mất tính tổng quát ta giả thiết có tập $\mathbf{o}$ đã sắp xếp theo $c(o_1) \geq c(o_2) \geq \ldots \geq c(o_m)$. Ta cũng giả sử rằng $\mathbf{s}'$ gồm T bộ cuối trong $\mathbf{s}$, có nghĩa là $\mathbf{s}' = \mathbf{s}_T$. Đặt $Q = t - T$, chúng ta có $\mathbf{s} = \mathbf{s}^Q \sqcup \mathbf{s}'$.

Các bổ đề dưới đây sẽ liên kết giữa lời giải dự tuyển $\mathbf{s}$ với $\mathbf{o}_2$. bổ đề đầu tiên, chỉ ra mối quan hệ giữa lời giải tối ưu $\mathbf{o}_2$ và $\mathbf{s}^j$ là lời giải dự tuyển có được sau khi thêm j phần tử bất kỳ.

Bổ đề 2.1. $f(\mathbf{o}_2) - f(\mathbf{o}_2^j) \leq f(\mathbf{s}^j)$ *với mọi* $0 \leq j \leq t$.

Chứng minh. Chúng ta có: $\mathbf{o}_2^0 = (\mathbf{o}_2 \sqcup \mathbf{s}^0) \sqcup \mathbf{s}^0$. Vì vậy, $f(\mathbf{o}_2) = f(\mathbf{o}_2^0)$. Với mọi $0 \leq j \leq t$, ta có:

$$f(\mathbf{o}_2) - f(\mathbf{o}_2^j) = \sum_{i=1}^{j} (f(\mathbf{o}_2^{i-1}) - f(\mathbf{o}_2^i)) \tag{2.3}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{o}_2^{i-1}) - f(\mathbf{o}_2^{i-1/2})) \tag{2.4}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{s}^{i-1/2}) - f(\mathbf{s}^{i-1})) \tag{2.5}$$

$$\leq \sum_{i=1}^{j} (f(\mathbf{s}^{i}) - f(\mathbf{s}^{i-1})) \tag{2.6}$$

$$\leq f(\mathbf{s}^j). \tag{2.7}$$

Trong đó bất đẳng thức (2.4) xảy ra do tính đơn điệu của f, bất đẳng thức (2.5) do tính k-submodular của f và bất đẳng thức (2.6) là do cách lựa chọn các phần tử của thuật toán. □

Do lời giải cuối cùng lấy các phần tử cuối của lời giải dự tuyển, ta cần tìm mối quan hệ giữa $\mathbf{s}'$ và $\mathbf{s}^t$. Ta có bổ đề dưới đây.

Bổ đề 2.2. $f(\mathbf{s}') \geq f(\mathbf{s}^t)/3$.

Chứng minh. Nếu $c(\mathbf{s}) \leq B$, $\mathbf{s}' = \mathbf{s}$ thì bổ đề vẫn đúng. Vì vậy, chúng ta chỉ cần xét trường hợp $c(\mathbf{s}) > B$. Ta có:

$$f(\mathbf{s}) - f(\mathbf{s}^Q) = \sum_{j=Q+1}^{t} \Delta_{(e_j,i_j)} f(\mathbf{s}^{j-1}) \tag{2.8}$$

$$\geq \sum_{j=Q+1}^{t} c(e_j) \frac{f(\mathbf{s}^{j-1})}{B} \tag{2.9}$$

$$\geq \sum_{j=Q+1}^{t} c(e_j) \frac{f(\mathbf{s}^{Q})}{B} \tag{2.10}$$

$$\geq c(\mathbf{s}') \frac{f(\mathbf{s}^{Q})}{B}. \tag{2.11}$$

Bất đẳng thức (2.9) xảy ra do sự lựa chọn bộ (e, i'_e) vào tập $\mathbf{s}$ ở dòng 6 của Thuật toán 2, bất đẳng thức (2.10) là do tính đơn điệu của hàm f.

Vì $\mathbf{s}'$ được lựa chọn từ $\mathbf{s}^t$ để tổng chi phí của $\mathbf{s}$ gần B nhất và mỗi phần tử $e \in supp(\mathbf{s})$ có chi phí nhiều nhất là $B/2$ nên

$$c(\mathbf{s}') > B - B/2 \geq B/2.$$

Suy ra $f(\mathbf{s}) - f(\mathbf{s}^Q) \geq f(\mathbf{s}^Q)/2$. Vì vậy $f(\mathbf{s}^Q) \leq 2f(\mathbf{s}^t)/3$. Mặt khác, do tính chất k-submodular của hàm f chúng ta có: $f(\mathbf{s}^t) \leq f(\mathbf{s}^Q) + f(\mathbf{s}')$. Nên:

$$f(\mathbf{s}') \geq f(\mathbf{s}^t) - f(\mathbf{s}^Q) \geq \frac{f(\mathbf{s}^t)}{3}. \tag{2.12}$$

Vậy bổ đề đã được chứng minh. □

Hai bổ đề tiếp theo phân tích mối quan hệ giữa lời giải tối ưu trên tập V_2 và $\mathbf{s}^t$.

Bổ đề 2.3. $f(\mathbf{o}_2^t) \leq 2f(\mathbf{s}^t)$.

Chứng minh. Từ định nghĩa của $\mathbf{u}^t$, có các phần tử trong $\mathbf{u}^t$ sẽ không thỏa mãn điều kiện ở dòng 7 Thuật toán 2 và $c(\mathbf{u}^t) \leq c(\mathbf{o}_2) \leq B$. Đặt $\mathbf{s}^{<u_j}$ là tập $\mathbf{s}$ ngay

trước khi phần tử u_j được truy xuất $(j < r)$. Ta có:

$$f(\mathbf{o}_2^t) - f(\mathbf{s}^t) = f(\mathbf{u}^t \sqcup \mathbf{s}^t) - f(\mathbf{s}^t) = \sum_{j=1}^{r} (f(\mathbf{u}_j^t) - f(\mathbf{u}_{j-1}^t))$$

$$\leq \sum_{j=1}^{r} (f(\mathbf{s}^{<u_j} \sqcup (u_j, i_j)) - f(\mathbf{s}^{<u_j}) \tag{2.13}$$

$$\leq \sum_{j=1}^{r} \Delta_{(u_j,i_j)} f(\mathbf{s}^{<u_j}) \tag{2.14}$$

$$< \sum_{j=1}^{r} c(u_j) \frac{f(\mathbf{s}^{<u_j})}{B} \tag{2.15}$$

$$\leq \sum_{j=1}^{r} c(u_j) \frac{f(\mathbf{s}^t)}{B} \tag{2.16}$$

$$\leq c(\mathbf{u}^t) \frac{f(\mathbf{s}^t)}{B} \leq f(\mathbf{s}^t). \tag{2.17}$$

Trong đó, bất đẳng thức (2.13) là do tính chất k-submodular của hàm f, bất đẳng thức (2.14) là do định nghĩa của $\mathbf{s}^{<u_j}$, bất đẳng thức (2.15) do sự lựa chọn của thuật toán và bất đẳng thức (2.16) là do tính chất đơn điệu của hàm f. Vì vậy, chúng ta có: $f(\mathbf{o}_2^t) \leq 2f(\mathbf{s}^t)$. □

Bổ đề 2.4. $f(\mathbf{s}') \geq f(\mathbf{o}_2)/9$.

Chứng minh. Áp dụng Bổ đề 2.1, 2.3, với $j = t$, ta có:

$$f(\mathbf{o}_2) - f(\mathbf{s}^t) = f(\mathbf{o}_2) - f(\mathbf{o}_2^t) + f(\mathbf{o}_2^t) - f(\mathbf{s}^t) \tag{2.18}$$

$$\leq f(\mathbf{s}^t) + f(\mathbf{o}_2^t) - f(\mathbf{s}^t) = f(\mathbf{o}_2^t) \tag{2.19}$$

$$\leq 2f(\mathbf{s}^t). \tag{2.20}$$

Vậy $f(\mathbf{s}^t) \geq f(\mathbf{o}_2)/3$. Kết hợp với Bổ đề 2.2, ta có $f(\mathbf{s}') \geq f(\mathbf{o}_2)/9$. □

Từ đó suy ra mối quan hệ giữa lời giải cuối cùng của thuật toán $\mathbf{s}_{final}$ và lời giải tối ưu $\mathbf{o}$ qua định lý về đảm bảo lý thuyết của thuật toán dưới đây.

Định lý 2.2. *Thuật toán* **FA** *là thuật toán luồng 1 lần quét cho tỉ lệ xấp xỉ là* $1/10$ *và tốn* nk *truy vấn.*

Chứng minh. Thuật toán chỉ cần một lần quét qua tập cơ sở và mỗi phần tử e cần k truy vấn để tìm vị trí i_e. Do đó lượng truy vấn cần là nk. Phần tiếp theo chứng

minh tỉ lệ xấp xỉ của thuật toán. Ta có:

$$f(\mathbf{o}) \leq f(\mathbf{o}'_1) + f(\mathbf{o}'_2) \tag{2.21}$$
$$\leq f(\mathbf{o}_1) + f(\mathbf{o}_2) \tag{2.22}$$
$$\leq f((e_m, i_m)) + 9f(\mathbf{s}') \leq 10f(\mathbf{s}_{final}). \tag{2.23}$$

Trong đó (2.21) do tính k-submodular của hàm f, các bất đẳng thức (2.22) và (2.23) do định nghĩa của $\mathbf{o}_1$, $\mathbf{o}_2$ và (e_m, i_m). □

2.5.3. Thuật toán xấp xỉ nhanh cải tiến: IFA

Tiếp theo, luận án trình bày thuật toán xấp xỉ nhanh cải tiến **IFA** (*Improved Fast Approximation*). Thuật toán này nâng tỉ lệ xấp xỉ lên $(1/4 - \epsilon)$ với độ phức tạp truy vấn là $O(kn/\epsilon)$. Ý tưởng chính của **IFA** là sử dụng lời giải của **FA** để cung cấp một khoảng giới hạn của giá trị tối ưu opt và tích hợp với ngưỡng tham lam để phát triển tỉ lệ xấp xỉ của thuật toán bằng cách tạo ra $O(1/\epsilon)$ lần quét tập cơ sở. Chi tiết của thuật toán được giới thiệu trong Thuật toán 3 dưới đây.

Algorithm 3 IFA

Input: V, f, k, $B > 0$, $\epsilon > 0$.
Output: A solution $\mathbf{s}$.

1: $\mathbf{s}_{max} \leftarrow \mathbf{FA}(V, f, k, B)$; $\Gamma \leftarrow f(\mathbf{s}_{max})$
2: $S \leftarrow \{i : i \in \mathbb{N}, \Gamma \leq (1+\epsilon)^i \leq 10\Gamma\}$, $\mathbf{s}_v \leftarrow \mathbf{0} \forall v \in S$
3: **for all** $e \in V$ **do**
4: **for all** $v \in S$ **do**
5: $i_v \leftarrow \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s}_v)$
6: $\theta_v = v/(2B)$
7: **if** $c(\mathbf{s}_v) + c(e) \leq B$ **and** $\Delta_{(e,i_v)} f(\mathbf{s}_v)/c(e) \geq \theta_v$ **then**
8: $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
9: **end if**
10: **end for**
11: **end for**
12: $\mathbf{s}_{final} \leftarrow \arg\max_{\mathbf{s}' \in \{\mathbf{s}_{max}, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{|S|}\}} f(\mathbf{s}')$
13: **return** $\mathbf{s}_{final}$

IFA cũng nhận bộ (V, f, k, B) của kSMK và tham số chính xác $\epsilon > 0$ làm đầu vào. Ý tưởng chính của **IFA** là sử dụng lời giải $\mathbf{s}_{max}$ của **FA** như là một chu trình con để tạo ra một vùng giới hạn cho lời giải tối ưu (dòng 1). Từ Định lý 2.2, ta có $\Gamma \leq \mathsf{opt} \leq 10\Gamma$. Giới hạn này giúp ta xây dựng tập S, ứng với mỗi $v \in S$, ta xây dựng các lời giải dự tuyển $\mathbf{s}_v$.

Thuật toán bao gồm hai vòng lặp là vòng lặp ngoài và vòng lặp bên trong nó. Vòng ngoài quét mỗi phần tử e trong tập cơ sở V và vòng lặp bên trong xem xét mỗi lời giải ứng viên $\mathbf{s}_v$ ứng với từng phần tử v được lọc ra từ tập S. Dựa theo Định lý 2.2, thuật toán này dựng tập S để giới hạn lại số lượng các lời giải ứng viên $\mathbf{s}_v$. Chúng ta đặt (e, i_v) là bộ cho lợi nhuận biên lớn nhất khi thêm vào tập $\mathbf{s}_v$.

Khi một phần tử e đến, thuật toán sẽ xử lý các vấn đề sau: (1) chọn vị trí i_v cho lợi nhuận biên lớn nhất ứng với $\mathbf{s}_v$ và e (dòng (5)); (2) sử dụng ngưỡng tham lam $\theta_v = v/(2B)$ để thêm một phần tử e vào trong $\mathbf{s}_v$ nếu nó có mật độ tăng thêm vượt qua ngưỡng này mà không vi phạm ràng buộc về ngân sách (dòng (7)).

Để xây dựng thuật toán và phân tích các tỉ lệ xấp xỉ, độ phức tạp truy vấn, thuật toán này cần sử dụng bộ ký hiệu của riêng thuật toán như sau:

- $\mathbf{s}_v = \{(e_1, i_1), (e_2, i_2), \ldots, (e_q, i_q)\}$ là lời giải dự tuyển ứng với các phần tử $v \in S$ sau khi kết thúc vòng lặp bên ngoài.
- $\mathbf{s}_v^j = \{(e_1, i_1), (e_2, i_2), \ldots, (e_j, i_j)\}, 1 \leq j \leq q$ là tập $\mathbf{s}_v$ sau khi thêm j phần tử. Như vậy $\mathbf{s}_v^0 = \mathbf{0}$.
- $\mathbf{s}_v^{<e}$ là $\mathbf{s}_v$ ngay trước khi e được xử lý.
- $\mathbf{u} = \{(u_1, i_1), (u_2, i_2), \ldots, (u_r, i_r)\}$ là tập các phần tử thuộc tập $\mathbf{o}$ nhưng không nằm trong $\mathbf{s}_v$, $r = |supp(\mathbf{u})|$.
- $\mathbf{u}_l = \mathbf{s}_v \sqcup \{(u_1, i_1), (u_2, i_2), \ldots, (u_l, i_l)\}, \forall 1 \leq l \leq r$ và $\mathbf{u}_0 = \mathbf{s}_v$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}_v^j) \sqcup \mathbf{s}_v^j$. $\mathbf{o}^j$ bao gồm các phần tử của tập lời giải tối ưu mà có mặt trong $\mathbf{s}_v^j$ và cả các phần tử còn lại nằm trong $\mathbf{s}_v^j$.
- $\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}_v^j) \sqcup \mathbf{s}_v^{j-1}$, là bao gồm các phần tử trong $\mathbf{o}$ đã xuất hiện trong $\mathbf{o}^j$ nhưng chưa xuất hiện trong $\mathbf{o}^{j-1}$.

Ta có các bổ đề phân tích mối quan hệ giữa các tập $\mathbf{o}$ và $\mathbf{s}_v$ dưới đây.

Bổ đề 2.5. *Với bất kỳ $v \in S$, nếu không có bất kỳ phần tử $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ để $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o})/c(o) \geq \theta_v$ và $c(\mathbf{s}_v^{<o}) + c(o) > B$, chúng ta có: $f(\mathbf{o}) \leq 2f(\mathbf{s}_v) + c(\mathbf{o})\theta_v$.*

Chứng minh. Quy luật chọn các (e, i_v) của thuật toán **IFA** là giống với chọn (e, i'_e) của thuật toán **FA**, chúng ta có kết quả tương tự như ở Bổ đề 2.1, có nghĩa là

$f(\mathbf{o}) - f(\mathbf{o}^q) \leq f(\mathbf{s}_v)$. Vì vậy:

$$f(\mathbf{o}) - f(\mathbf{s}_v) = f(\mathbf{o}) - f(\mathbf{o}^q) + f(\mathbf{o}^q) - f(\mathbf{s}_v) \tag{2.24}$$

$$\leq f(\mathbf{s}_v) + \sum_{j=1}^{r} (f(\mathbf{u}_j) - f(\mathbf{u}_{j-1})) \tag{2.25}$$

$$\leq f(\mathbf{s}_v) + \sum_{j=1}^{r} \Delta_{(u_j, i_j)} f(\mathbf{s}_v^{<u_j}) \tag{2.26}$$

$$\leq f(\mathbf{s}_v) + \sum_{j=1}^{r} c(u_j)\theta_v \tag{2.27}$$

$$\leq f(\mathbf{s}_v) + c(\mathbf{o})\theta_v. \tag{2.28}$$

Trong đó, bất đẳng thức (2.26) do tính chất k-submodular, bất đẳng thức (2.27) là xuất phát từ định nghĩa của $\mathbf{s}_v^{<o}$, và bất đẳng thức (2.28) xuất phát từ định nghĩa của $\mathbf{u}$ và $\mathbf{o}$. Từ đó, ta có điều phải chứng minh. □

Cuối cùng, ta xây dựng được các đảm bảo lý thuyết như định lý dưới đây.

Định lý 2.3. *Với $0 < \epsilon < 1/4$, thuật toán* **IFA** *là thuật toán xấp xỉ với tỉ lệ là $1/4 - \epsilon$, cần $O(nk/\epsilon)$ truy vấn.*

Chứng minh. Thuật toán cần nk truy vấn để gọi thuật toán **FA** và chỉ cần 1 lần quét tập cơ sở để kết thúc vòng lặp ngoài (các dòng 3 đến 10). Với mỗi một phần tử đang xét, nó cần nhiều nhất là $k \cdot \lceil \log_{(1+\epsilon)}(10) \rceil$ truy vấn để cập nhật các $\mathbf{s}_v, v \in S$. Kết hợp các trường hợp, số lượng truy vấn cần nhiều nhất là:

$$\begin{aligned} nk + nk\lceil \log_{(1+\epsilon)}(10) \rceil &\leq nk + nk(1 + \log_{(1+\epsilon)}(10)) = 2nk + nk\frac{\ln(10)}{\ln(1+\epsilon)} \\ &\leq 2nk + nk\frac{\ln(10)}{\ln \frac{1}{1-\frac{\epsilon}{2}}} = 2nk - nk\frac{\ln(10)}{\ln(1-\frac{\epsilon}{2})} \\ &\leq 2nk + \frac{2}{\epsilon} nk \ln(10) = O(\frac{nk}{\epsilon}). \end{aligned}$$

Trong đó, bất đẳng thức thứ hai là do $1 + \epsilon \geq \frac{1}{1-\frac{\epsilon}{2}}$, với $\epsilon \in (0, 1)$ và bất đẳng thức thứ ba là từ bất đẳng thức $\ln(x+1) \leq x$, với mọi $x \in (-1, 0)$.

Tiếp theo, từ Định lý 2.2, ta có $\Gamma \leq \mathsf{opt} \leq 10\Gamma$. Vì vậy, sẽ có một số nguyên

$v \in S$ để $\mathsf{opt}/(1+\epsilon) \leq v < \mathsf{opt}$. Từ đó, ta có:

$$f(\mathbf{s}_v^j) = \sum_{i=1}^{j}(f(\mathbf{s}_v^i) - f(\mathbf{s}_v^{i-1})) \geq \sum_{i=1}^{j} c(e_i)\theta_v = c(\mathbf{s}_v^j)\theta_v. \tag{2.29}$$

Ta xét 2 trường hợp sau:

- Trường hợp 1. Sẽ tồn tại một phần tử $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ sao cho $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) \geq c(o) \cdot \theta_v$ và $c(\mathbf{s}_v^{<o}) + c(o) > B$. Nhắc lại, đặt $(e_m, i_m) = \arg\max_{e \in V, i \in [k]} f((e,i))$, có:

$$\begin{aligned}
f(\mathbf{s}_{final}) &\geq \max\{f(\mathbf{s}_v), f((e_m, i_m))\} && (2.30)\\
&\geq \max\{f(\mathbf{s}_v^{<o}), f((o, \mathbf{o}(o)))\} && (2.31)\\
&\geq \frac{f(\mathbf{s}_v^{<o}) + f((o, \mathbf{o}(o)))}{2} && (2.32)\\
&\geq \frac{f(\mathbf{s}_v^{<o} \sqcup (o, \mathbf{o}(o))}{2} && (2.33)\\
&= \frac{\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) + f(\mathbf{s}_v^{<o})}{2} && (2.34)\\
&\geq \frac{\theta_v c(o) + \theta_v c(\mathbf{s}_v^{<o})}{2} \geq \frac{B\theta_v}{2} \geq \frac{\mathsf{opt}}{4(1+\epsilon)} && (2.35)\\
&\geq (\frac{1}{4} - \epsilon)\mathsf{opt}. && (2.36)
\end{aligned}$$

- Trường hợp 2. Không có phần tử $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ nào như vậy. Bằng Bổ đề 2.5, ta có:

$$f(\mathbf{o}) \leq 2f(\mathbf{s}_v) + B\theta_v \leq 2f(\mathbf{s}_v) + \frac{\mathsf{opt}}{2}. \tag{2.37}$$

Do vậy $f(\mathbf{s}_{final}) \geq f(\mathbf{s}_v) \geq \mathsf{opt}/4$. Kết hợp tất cả các trường hợp trên, ta có điều phải chứng minh. □

2.5.4. Thuật toán xấp xỉ tăng cường: IFA+

Để cải tiến các thuật toán trước, thuật toán tiếp theo được đề xuất là thuật toán xấp xỉ tăng cường **IFA+** (*Improved Fast Approximation Plus*), Thuật toán 4. Thuật toán này cải tiến tỉ lệ xấp xỉ của **IFA** mà vẫn đảm bảo trong thời gian tuyến tính. **IFA+** cho tỉ lệ xấp xỉ tất định là $1/3 - \epsilon$. Ý tưởng của thuật toán này dựa trên giảm dần ngưỡng nhằm bổ sung thêm các phần tử tốt từ tập V vào lời giải dự tuyển $\mathbf{s}$. Sau đó, phân hoạch lại $\mathbf{s}$ để lấy thêm các phần tử tốt hơn từ V đưa vào các phân hoạch của $\mathbf{s}$ trong khi tổng chi phí vẫn còn nhỏ hơn ngân sách. Ý tưởng này

xuất phát từ nhận xét khi ta chọn được một tập lời giải dự tuyển thì các phần tử tốt vẫn còn nằm ở trong tập V. Do vậy, ta có thể tăng cường chất lượng lời giải của $\mathbf{s}$.

Thuật toán **IFA+** bao gồm 2 pha. Pha đầu tiên (dòng 2) sử dụng cách thiết kế tương tự như **IFA** nhưng có một vài điều chỉnh. Pha đầu tiên này cũng gọi **FA** như là một chu trình con để có được khoảng bao của giá trị tối ưu trong đoạn $[\Gamma, 10\Gamma]$ (dòng 3). Sau đó, nó sử dụng ngưỡng tham lam để thêm các phần tử nào có mật độ tăng thêm cao vào trong tập lời giải dự tuyển $\mathbf{s}$.

Cụ thể, pha này sẽ bao gồm một số bước lặp, mỗi bước quét tập cơ sở 1 lần (dòng 4-12). Với một phần tử e đang xét, thuật toán sẽ tìm vị trí tốt nhất $i_e = \arg\max_{i\in[k]} \Delta_{(e,i)} f(\mathbf{s})$ và e được thêm vào tập $\mathbf{s}$ nếu mật độ tăng thêm của nó $\Delta_{(e,i)} f(\mathbf{s})/c(e)$ lớn hơn hoặc bằng ngưỡng θ mà không vi phạm giới hạn ngân sách. Ngưỡng θ được khởi tạo bằng $10\Gamma/(3\epsilon B)$ và giảm dần theo hệ số $(1-\epsilon)$ sau từng bước lặp cho đến khi nhỏ hơn $\Gamma(1-\epsilon)/(3B)$.

Pha thứ hai (dòng 12-23) được sử dụng để tăng cường tỉ lệ xấp xỉ của $\mathbf{s}$. Ý tưởng then chốt của pha này là tiếp tục khám phá trong tập $\mathbf{s}$ để tăng cường chất lượng của $\mathbf{s}^T$, với $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$ là k-tập $\mathbf{s}$ sau khi thêm j phần tử $1 \le j \le t$, $\mathbf{s}^0 = \mathbf{0}$ và $T = \max\{j \in \mathbb{N} : \mathbf{s}^j + c(o_1) \le B\}$.

Để tìm ra $\mathbf{s}^T$, thuật toán tìm rất nhiều các tập dự đoán $\mathbf{s}^q$ với tổng chi phí nhiều nhất là l (dòng (16)) với l sẽ được tăng dần dần từ ϵB tới $(1+\epsilon)B$. Thuật toán sau đó sẽ quét lại tập cơ sở V để thêm phần tử e_l mới tốt nhất từ V vào trong $\mathbf{s}'_l$ mà không vi phạm ràng buộc về ngân sách (dòng (20)).

Cuối cùng, lời giải cuối sẽ chọn giá trị lớn nhất giữa $\mathbf{s}_{max}, \mathbf{s}$, và danh sách các $\mathbf{s}_{(j)}$ với $j \in [l]$ (dòng 24). Chi tiết của thuật toán được trình bày ở thuật toán 4.

Tương tự như các thuật toán trên, để phân tích cho thuật toán **IFA+**, ta sử dụng các quy ước ký hiệu sau:

- $\mathbf{s} = \{(e_1, i_1), \ldots, (e_t, i_t)\}$ là k-tập $\mathbf{s}$ sau khi kết thúc vòng lặp đầu tiên, $t = |supp(\mathbf{s})|$.
- $\mathbf{s}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: là k-tập $\mathbf{s}$ sau khi thêm j phần tử $1 \le j \le t$, như vậy $\mathbf{s}^0 = \mathbf{0}$, $\mathbf{s}^t = \mathbf{s}$.
- $\mathbf{s}_j$: k-tập $\mathbf{s}$ sau khi kết thúc bước lặp thứ j của vòng lặp đầu tiên.
- $\mathbf{s}_j^i$: là k-tập chứa i phần tử đầu tiên trong $\mathbf{s}_j$.
- $T = \max\{j \in \mathbb{N} : \mathbf{s}^j + c(o_1) \le B\}$, trong đó $o_1 = \max_{o \in supp(\mathbf{o})} c(o)$.
- $\mathbf{o}' = \{(o_2, i_2^*), \ldots, (o_m, i_m^*)\}$, là lời giải tối ưu sau khi chọn được (o_1, i_1^*).
- θ_j: ngưỡng θ tại bước lặp thứ j của vòng lặp đầu tiên.
- $\theta_{(j)}$: ngưỡng θ khi bộ (e_j, i_j) được thêm vào $\mathbf{s}$ trong vòng lặp đầu tiên.

Phần tiếp theo sẽ phân tích Bổ đề 2.6. Đây là bổ đề đơn giản nhưng đóng vai trò quan trọng không thể thiếu để phân tích tỉ lệ xấp xỉ.

Algorithm 4 IFA+

Input: V, f, k, $B > 0$, $\epsilon > 0$.
Output: Output: A solution $\mathbf{s}$.

\\ Phase 1
$\mathbf{s}_{max} \leftarrow \mathbf{FA}(V, f, k, B)$, $\mathbf{s} \leftarrow \mathbf{0}$;
$\Gamma \leftarrow f(\mathbf{s}_{max})$, $\theta \leftarrow 10\Gamma/(3\epsilon B)$
while $\theta \geq (1-\epsilon)\Gamma/(3B)$ **do**
  for all $e \in V \setminus supp(\mathbf{s})$ **do**
    $i_e \leftarrow \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s})$
    if $c(\mathbf{s}) + c(e) \leq B$ **and** $\Delta_{(e,i_e)} f(\mathbf{s})/c(e) \geq \theta$ **then**
      $\mathbf{s} \leftarrow \mathbf{s} \sqcup (e, i_e)$
    end if
  end for
  $\theta \leftarrow (1-\epsilon)\theta$
end while
\\ Phase 2: Boosting quality of solutions
$l \leftarrow \epsilon B$
while $l \leq B$ **do**
  Find $q \leftarrow \max\{i : i \leq |supp(\mathbf{s})|, c(\mathbf{s}^i) \leq l\}$
  $\mathbf{s}'_l \leftarrow \mathbf{s}^q$
  if $\mathbf{s}'_l \neq \mathbf{s}_{(l/(1+\epsilon))}$ **then**
    $(e_l, i_l) \leftarrow \arg\max_{i \in [k], e \in V \setminus supp(\mathbf{s}'_l): c(\mathbf{s}'_l)+c(e) \leq B} \Delta_{(e,i)} f(\mathbf{s}'_l)$
    $\mathbf{s}_{(l)} \leftarrow \mathbf{s}'_l \sqcup (e_l, i_l)$
  end if
  $l \leftarrow (1+\epsilon)l$
end while
$\mathbf{s} \leftarrow \arg\max_{\mathbf{s}'' \in \{\mathbf{s}_{max}, \mathbf{s}, \mathbf{s}_{(\epsilon B)}, \mathbf{s}_{(\epsilon B(1+\epsilon))}, \ldots, \mathbf{s}_{(l)}} f(\mathbf{s}'')$
return $\mathbf{s}$

Bổ đề 2.6. *Với bất kỳ k-tập $\mathbf{x}$ nào ta đều có:*

$$f(\mathbf{x}) \leq 2f(\mathbf{s}_j) + \sum_{e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)} \Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j).$$

Nếu $\max_{e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)} c(e) + c(\mathbf{s}_j) \leq B$, *chúng ta sẽ có:*

$$f(\mathbf{x}) < 2f(\mathbf{s}_j) + c(\mathbf{x})\theta_j.$$

Chứng minh. Sử dụng lập luận tương tự Bổ đề 2.5, chúng ta cũng sẽ có:

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{s}_j) &= f(\mathbf{x}) - f(\mathbf{x}^i) + f(\mathbf{x}^i) - f(\mathbf{s}_j) \\ &\leq f(\mathbf{s}_j) + \sum_{e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)} \Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j). \end{aligned}$$

Với $e \notin supp(\mathbf{s}_j)$ bất kỳ mà không vi phạm ràng buộc tổng lực lượng, có nghĩa là $c(e) + c(\mathbf{s}_j) \leq B$, nó sẽ không vượt qua được điều kiện ở dòng 7 của Thuật toán 4 tại vòng lặp j của vòng lặp đầu tiên, với $\mathbf{s}^{<e}$ là $\mathbf{s}$ ngay trước khi e được xử lý. Suy ra:

$$\frac{\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j)}{c(e)} \leq \frac{\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}^{<e})}{c(e)} < \theta_j.$$

Vì vậy, nếu $\max_{e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)} c(e) + c(\mathbf{s}_j) \leq B$, thì $\Delta_{(e,\mathbf{x}(e))} f(\mathbf{s}_j) < c(e)\theta_j$ với mọi $e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)$, nên có:

$$f(\mathbf{x}) - f(\mathbf{s}_j) < f(\mathbf{s}_j) + \sum_{e \in supp(\mathbf{x}) \setminus supp(\mathbf{s}_j)} c(e)\theta_j \leq f(\mathbf{s}_j) + c(\mathbf{x})\theta_j.$$

Suy ra điều phải chứng minh. □

Các bổ đề tiếp theo phân tích được tỉ lệ xấp xỉ của thuật toán dựa vào điều kiện của $c(o_1)$.

Bổ đề 2.7. *Nếu* $c(o_1) > (1-\epsilon)B$, $f(\mathbf{s}) \geq (\frac{1}{3} - \epsilon)\mathsf{opt}$.

Chứng minh. Theo định nghĩa của $\mathbf{o}'$, chúng ta có $c(\mathbf{o}') \leq B - c(o_1) \leq \epsilon B$. Ngưỡng θ giảm dần từ $\frac{10\Gamma}{3\epsilon B}$ xuống $\frac{(1-\epsilon)\Gamma}{3B}$ theo hệ số $1-\epsilon$ sau mỗi bước lặp của vòng lặp đầu tiên. Vì vậy, số bước lặp của vòng lặp đầu tiên nhiều nhất là:

$$\lceil \log_{(1-\epsilon)}(\frac{\epsilon(1-\epsilon)}{10}) \rceil \leq 2 + \frac{\ln(\epsilon/10)}{\ln(1-\epsilon)} = 2 - \frac{\ln(10/\epsilon)}{\ln(1-\epsilon)} \leq 2 + \frac{\ln(10/\epsilon)}{\epsilon} \quad (2.38)$$

Trong đó, bất đẳng thức đầu tiên là vì ta có bất đẳng thức $x \geq \ln(1+x)$, với mọi $x \in (-1, 0)$. Xem xét bước lặp $j = \lceil \log_{(1-\epsilon)}(\frac{\epsilon \mathsf{opt}}{10\Gamma}) \rceil$, ta có:

$$\frac{(1-\epsilon)\mathsf{opt}}{3B} < \theta_j = \frac{10(1-\epsilon)^j \Gamma}{3\epsilon B} \leq \frac{\mathsf{opt}}{3B}$$

Ta xem xét thời điểm sau khi bước lặp j kết thúc, có 2 trường hợp sau xảy ra:
-Trường hợp 1: Nếu $supp(\mathbf{s}^T) \subseteq supp(\mathbf{s}_j)$, có nghĩa là thuật toán đạt được $\mathbf{s}^T$ trước khi kết thúc bước lặp j.

+/ Nếu $c(\mathbf{s}_j) < (1-\epsilon)B$. Từ tính chất k-submodular của f, có $f(\mathbf{o}') \geq f(\mathbf{o}) - f((o_1, i_1^*))$. Với bất kì phần tử $e \in supp(\mathbf{o}') \backslash supp(\mathbf{s}_j)$, ta đều có $c(\mathbf{s}_j) + \max_{e \in supp(\mathbf{o}')} c(e) \leq c(\mathbf{s}_j) + c(\mathbf{o}') < B$. Áp dụng Bổ đề 2.6, ta có được $f(\mathbf{o}') \leq 2f(\mathbf{s}_j) + c(\mathbf{o}')\theta_j = 2f(\mathbf{s}_j) + \epsilon\mathsf{opt}/3$. Vì vậy,

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{s}) &\leq f(\mathbf{o}) - f((o_1, i_1^*)) && (2.39)\\ &\leq f(\mathbf{o}') && (2.40)\\ &\leq 2f(\mathbf{s}_j) + \frac{\epsilon}{3}\mathsf{opt} && (2.41)\\ &\leq 2f(\mathbf{s}) + \frac{\epsilon}{3}\mathsf{opt}. && (2.42) \end{aligned}$$

Trong đó, bất đẳng thức (2.39), (2.42) là do cách chọn lời giải cuối của thuật toán và bất đẳng thức (2.40) là do tính k-submodular của f. Vì vậy $f(\mathbf{s}) \geq (\frac{1}{3} - \frac{\epsilon}{9})\mathsf{opt}$.
+/ Nếu $c(\mathbf{s}_j) \geq (1-\epsilon)B$, ta có:

$$f(\mathbf{s}_j) \geq c(\mathbf{s}_j)\theta_j = \frac{(1-\epsilon)^2\mathsf{opt}}{3} > (\frac{1}{3} - \epsilon)\mathsf{opt}.$$

- Trường hợp 2: Nếu $supp(\mathbf{s}_j) \subset supp(\mathbf{s}^T)$. Từ định nghĩa của T, ta áp dụng Bổ đề 2.6 với lưu ý rằng $\theta_{(T)} \leq \theta_j$, có được:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{s}^T) &\leq f(\mathbf{s}^T) + c(\mathbf{o})\theta_{(T)}\\ &\leq f(\mathbf{s}^T) + c(\mathbf{o})\theta_j\\ &\leq f(\mathbf{s}^T) + \frac{\mathsf{opt}}{3}. \end{aligned}$$

Từ đó dẫn đến $f(\mathbf{s}) \geq f(\mathbf{s}^T) \geq \mathsf{opt}/3$. □

Bổ đề 2.8. *Nếu* $c(o_1) \leq (1-\epsilon)B$, $f(\mathbf{s}) \geq (\frac{1}{3} - \epsilon)\mathsf{opt}$.

Chứng minh. Nếu $\mathbf{s}^T = \mathbf{s}$ sau khi kết thúc vòng lặp chính, từ Bổ đề 2.6 dễ dàng chứng minh được rằng:

$$\begin{aligned} f(\mathbf{o}) - f(\mathbf{s}^T) &\leq f(\mathbf{s}^T) + c(\mathbf{o})\frac{\Gamma(1-\epsilon)}{3B}\\ &\leq f(\mathbf{s}^T) + \frac{(1-\epsilon)}{3}\mathsf{opt}. \end{aligned}$$

Vì vậy, $f(\mathbf{s}^T) > (1/3 + \epsilon/6)\mathsf{opt}$. Ta xem xét trường hợp còn lại khi $\mathbf{s}^T \neq \mathbf{s}$ sau khi

kết thúc vòng lặp chính. Lúc này $\mathbf{s}$ chứa ít nhất $T+1$ phần tử và $c(\mathbf{s}^{T+1})+c(o_1) \geq B$, suy ra $c(\mathbf{s}^{T+1}) \geq \epsilon B$. Xem xét bước lặp $j = \lceil \log_{1-\epsilon}(\mathsf{opt}/10\Gamma) \rceil$ của vòng lặp đầu tiên, ta có:

$$\frac{(1-\epsilon)\mathsf{opt}}{3\epsilon B} < \theta_j = \frac{10(1-\epsilon)^j\Gamma}{3\epsilon B} \leq \frac{\mathsf{opt}}{3\epsilon B}.$$

Tiếp theo, quan sát vòng lặp thứ hai của thuật toán. Vì l tăng dần từ ϵB đến B bởi hệ số $(1+\epsilon)$ sau mỗi bước lặp, tại bước lặp $h \geq 1$, ta có $l = \epsilon B(1+\epsilon)^{h-1}$. Vì $B - c(o_1) \geq \epsilon B$, nên tồn tại bước lặp h để

$$l = \epsilon B(1+\epsilon)^{h-1} \leq B - c(o_1) < \epsilon B(1+\epsilon)^h = l(1+\epsilon).$$

Sau bước lặp đó, do cách chọn $\mathbf{s}^q$ của thuật toán chúng ta có $c(\mathbf{s}^q) \leq l < c(\mathbf{s}^{q+1})$. Chúng ta xem xét các trường hợp sau đây:

-Trường hợp 1. Nếu thuật toán đạt được $\mathbf{s}^{q+1}$ bước lặp đầu tiên của vòng lặp đầu, ta có:

$$f(\mathbf{s}) \geq f(\mathbf{s}^{q+1}) \geq c(\mathbf{s}^{q+1})\theta_1 > \frac{B - c(o_1)}{1+\epsilon}\frac{\mathsf{opt}}{3\epsilon B} \geq \frac{\mathsf{opt}}{3(1+\epsilon)} \geq \frac{1}{3}(1-\epsilon)\mathsf{opt}.$$

-Trường hợp 2. Nếu thuật toán đạt được $\mathbf{s}^{q+1}$ sau bước lặp thứ $j > 1$ của vòng lặp đầu, đặt $\mathbf{s}^{<e}$ là $\mathbf{s}$ ngay trước khi e được xử lý. Mật độ tăng thêm của bất kỳ $e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q)$ đối với $\mathbf{s}^q$ là nhỏ hơn ngưỡng tại bước trước khi (e_{q+1}, i_{q+1}) được thêm vào $\mathbf{s}^q$, có nghĩa là:

$$\frac{\Delta_{(e,i_e)} f(\mathbf{s}^q)}{c(e)} \leq \frac{\theta_{(q+1)}}{1-\epsilon}.$$

Nếu $o_1 \notin supp(\mathbf{s}^q)$, bởi quy tắc chọn của $\mathbf{s}_{(l)}$ tại bước lặp đó và tính k-submodular của hàm f, chúng ta có $\mathbf{s}_{(l)} = \mathbf{s}'_l \sqcup (e_l, i_l) = \mathbf{s}^q \sqcup (e_l, i_l)$. Nhắc lại là $\mathbf{o}^q = (\mathbf{o} \sqcup \mathbf{s}^q) \sqcup \mathbf{s}^q$ với lưu ý rằng $o_1 \in supp(\mathbf{o}^q)$. Suy luận tương tự Bổ đề 2.1, ta cũng có

$f(\mathbf{o}) - f(\mathbf{o}^q) \leq f(\mathbf{s}^q)$. Vì vậy:

$$
\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) &\leq f(\mathbf{o}) - f(\mathbf{o}^q) + f(\mathbf{o}^q) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (2.43)\\
&\leq f(\mathbf{s}^q) + f(\mathbf{o}^q) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (2.44)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} \Delta_{(e, \mathbf{o}(e))} f(\mathbf{s}^q \sqcup (o_1, i_1^*)) && (2.45)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} \Delta_{(e, \mathbf{o}(e))} f(\mathbf{s}^q) && (2.46)\\
&\leq f(\mathbf{s}^q) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^q \sqcup (o_1, i_1^*))} c(e) \frac{\theta_{(q+1)}}{1-\epsilon} && (2.47)\\
&\leq f(\mathbf{s}^q) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon}. && (2.48)
\end{aligned}
$$

Trong đó bất đẳng thức (2.46) do tính k-submodular của f và bất đẳng thức (2.47) do áp dụng bất đẳng thức (2.43). Bằng áp dụng bất đẳng thức (2.48) và cách chọn lời giải cuối cùng ($f(\mathbf{s}) \geq f(\mathbf{s}_{(l)}) \geq f(\mathbf{s}^q)$), ta có:

$$
\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}_{(l)}) &= f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (e_l, i_l)) && (2.49)\\
&\leq f(\mathbf{o}) - f(\mathbf{s}^q \sqcup (o_1, i_1^*)) \ \text{(Do cách chọn } \mathbf{s}_{(l)}) && (2.50)\\
&\leq f(\mathbf{s}^q) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon} && (2.51)\\
&\leq f(\mathbf{s}) + \frac{(B - c(o_1))\theta_{(q+1)}}{1-\epsilon}. && (2.52)
\end{aligned}
$$

Sắp xếp lại bất đẳng thức (2.52), có được:

$$
\begin{aligned}
\theta_{(q+1)} &\geq (1-\epsilon) \frac{f(\mathbf{o}) - f(\mathbf{s}_{(l)}) - f(\mathbf{s}^{q+1})}{B - c(o_1)} && (2.53)\\
&\geq (1-\epsilon) \frac{f(\mathbf{o}) - 2f(\mathbf{s})}{B - c(o_1)}. && (2.54)
\end{aligned}
$$

Mặt khác, tại vòng lặp đầu, mật độ tăng thêm của mỗi phần tử được chọn ít nhất

bằng ngưỡng, nghĩa là $\Delta_{(e_j,i_j)}f(\mathbf{s}^{j-1})/c(e_j) \geq \theta_{(j)}$ với mọi $j = 1, \ldots, q+1$, nên:

$$f(\mathbf{s}) \geq f(\mathbf{s}^{q+1}) - f(\mathbf{s}^0) = \sum_{j=1}^{q+1} \Delta_{(e_j,i_j)}f(\mathbf{s}^{j-1}) \tag{2.55}$$

$$\geq \sum_{j=1}^{q+1} c(e_j)\theta_{(j)} \geq c(\mathbf{s}^{q+1})\theta_{(q+1)} \tag{2.56}$$

$$\geq \frac{B - c(o_1)}{1+\epsilon}\theta_{(q+1)} \text{ (Do bất đẳng thức (2.43))} \tag{2.57}$$

$$\geq \frac{1-\epsilon}{1+\epsilon}(f(\mathbf{o}) - 2f(\mathbf{s})). \tag{2.58}$$

Do đó, $f(\mathbf{s}) \geq (\frac{1}{3} - \frac{2\epsilon}{3(3-\epsilon)})f(\mathbf{o}) > (\frac{1}{3} - \epsilon)\mathsf{opt}$.
Nếu $o_1 \in supp(\mathbf{s}^q)$, ta cũng có được bất đẳng thức (2.52) bằng áp dụng các phép biến đổi sau:

$$\begin{aligned}
f(\mathbf{o}) - f(\mathbf{s}_{(l)}) &\leq f(\mathbf{o}) - f(\mathbf{s}^{q+1}) \\
&\leq f(\mathbf{s}^{q+1}) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^{q+1})} \Delta_{(e,\mathbf{o}(e))}f(\mathbf{s}^{<e}) \text{ (Áp dụng Bổ đề 2.6)} \\
&\leq f(\mathbf{s}^{q+1}) + \sum_{e \in supp(\mathbf{o}) \setminus supp(\mathbf{s}^{q+1})} c(e)\frac{\theta_{(q+1)}}{1-\epsilon} \\
&\leq f(\mathbf{s}^{q+1}) + \frac{\theta_{(q+1)}(B - c(o_1))}{1-\epsilon}.
\end{aligned}$$

Từ đây trở đi, bằng cách suy luận tương tự như trường hợp $o_1 \notin supp(\mathbf{s}^q)$, ta cũng có $f(\mathbf{s}) > (1/3 - \epsilon)\mathsf{opt}$. Kết hợp tất cả các trường hợp nêu trên, ta có điều phải chứng minh. □

Áp dụng các bổ đề trên, ta có định lý về đảm bảo lý thuyết của thuật toán.

Định lý 2.4. *Với $\epsilon \in (0, 1/3)$ bất kỳ, thuật toán* **IFA+** *có độ phức tạp truy vấn là $O(kn\log(1/\epsilon)/\epsilon)$ và trả về lời giải với tỉ lệ xấp xỉ là $1/3 - \epsilon$.*

Chứng minh. Thuật toán **IFA+** cần $O(nk)$ truy vấn để chạy thuật toán **FA**. Thuật toán **IFA+** gồm hai vòng lặp. Từ (2.38), vòng lặp đầu chứa ít nhất: $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}))$

bước lặp. Số bước lặp của vòng lặp thứ 2 nhiều nhất là:

$$\lceil \log_{(1+\epsilon)}(\frac{1}{\epsilon}) \rceil \leq 1 + \log_{(1+\epsilon)}(\frac{1}{\epsilon}) = 1 + \frac{\ln(\frac{1}{\epsilon})}{\ln(1+\epsilon)} \leq 1 + \frac{\ln(\frac{1}{\epsilon})}{\ln \frac{1}{1-\frac{\epsilon}{2}}}$$
$$= 1 - \frac{\ln(\frac{1}{\epsilon})}{\ln(1-\frac{\epsilon}{2})} \leq 1 + \frac{2}{\epsilon}\ln(\frac{1}{\epsilon}) = O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})).$$

Trong đó, bất đẳng thức đầu là do $1+\epsilon \geq \frac{1}{1-\frac{\epsilon}{2}}$, với mọi $\epsilon \in (0,1)$ và bất đẳng thức thứ hai là do áp dụng $\ln(1+x) \leq x$ với mọi $x \in (-1,0)$. Vì mỗi bước lặp của các vòng lặp trên quét 1 lần tập cơ sở và tốn mất kn truy vấn, ta có được số truy vấn nhiều nhất là:

$$O(nk) + 2 \cdot O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon})) \cdot kn = O(\frac{nk}{\epsilon}\log(\frac{1}{\epsilon})).$$

Tỉ lệ xấp xỉ của thuật toán tuân theo các Bổ đề 2.7, 2.8 bằng cách kết hợp 2 trường hợp sau: $c(o_1) \leq (1-\epsilon)B$ và $c(o_1) > (1-\epsilon)B$. Suy ra điều phải chứng minh. □

2.6. Các thuật toán cho trường hợp hàm mục tiêu không đơn điệu

2.6.1. Kết quả mới của luận án

Các bài toán ứng dụng của tối đa hàm submodular, hàm mục tiêu được chỉ ra là có thể không đơn điệu [55, 21, 97, 137]. Hệ quả tất yếu bài toán tối đa hàm k-submodular cũng có thể không đơn điệu. Pham và cộng sự [115] cũng đã giải quyết bài toán tổng quát tối đa hàm mục tiêu k-submodular với ngân sách giới hạn, hàm mục tiêu có thể không đơn điệu. Trong nghiên cứu của họ, các tác giả đưa ra lời giải xấp xỉ với độ phức tạp truy vấn gần tuyến tính (Bảng 2.4). Luận án tập trung giải quyết bài toán với hàm không đơn điệu bằng một thuật toán xấp xỉ cạnh tranh và giảm độ phức tạp truy vấn.

Bảng 2.4 đưa ra so sánh giữa các thuật toán được đề xuất, **LAA** và **RLA**, của luận án với các thuật toán tốt nhất hiện nay, thuật toán luồng tất định (DS) và thuật toán luồng ngẫu nhiên (RS) [115] đã đề cập đến ở phần trước.

Thuật toán	**Tỉ lệ xấp xỉ**	**Độ phức tạp truy vấn**	**Tính tất định**
DS [115]	$1/5-\epsilon$	$O(kn\log(n)/\epsilon)$	Có
RS [115]	$k/(5k-2)-\epsilon$	$O(kn\log(n)/\epsilon)$	Không
LAA (Thuật toán 5 trong luận án)	$1/19$	$O(kn)$	Có
RLA (Thuật toán 6 trong luận án)	$1/5-\epsilon$	$O(kn/\epsilon)$	Có

Bảng 2.4: So sánh các thuật toán cho kSMK không đơn điệu; Lưu ý DS và RS trong [115] là trường hợp đặc biệt khi $\beta = 1$.

Luận án cải tiến các thuật toán đã đề xuất với hàm mục tiêu đơn điệu, xây dựng được thuật toán cải tiến **RLA** (Thuật toán 6) giải cho không đơn điệu. **RLA** đạt được tỉ lệ xấp xỉ $1/5 - \epsilon$ và yêu cầu độ phức tạp của truy vấn $O(kn/\epsilon)$ trong đó $\epsilon > 0$ là tham số chính xác. Đây là thuật toán cho tỉ lệ xấp xỉ tương đương với tỉ lệ xấp xỉ tốt nhất hiện tại của thuật toán tất định cho kSMK không đơn điệu [115]. Phương pháp của **RLA** dựa trên *ngưỡng mật độ* để chọn ra các phần tử "tốt".

Đối với trường hợp này, luận án cũng tiến hành một số thử nghiệm tổng thể trên ứng dụng kIMK. Kết quả thử nghiệm cho thấy các thuật toán được đề xuất tiết kiệm truy vấn hơn so với thuật toán hiện đại nhất (được đề cập trong Bảng 2.4) và trả về các kết quả có thể so sánh được về mặt hiệu suất.

Bảng 2.4 so sánh các thuật toán trên ba khía cạnh tỉ lệ xấp xỉ, độ phức tạp của truy vấn và tính tất định hay không. Thuật toán **LAA** là phiên bản đơn giản, có tác dụng đưa số truy vấn về tuyến tính và giới hạn lại khoảng giá trị của opt để **RLA** sử dụng. Các trường nêu trong bảng cho thấy thuật toán được đề xuất có số lượng truy vấn thấp hơn và tỉ lệ xấp xỉ hằng số có giá trị tốt tương đương các thuật toán hiện đại.

2.6.2. Thuật toán xấp xỉ tuyến tính: LAA

Đầu tiên, thuật toán xấp xỉ tuyến tính **LAA** (*Linear Approximation Algorithm*) được tạo ra làm nền tảng để xây dựng thuật toán cải tiến. Thuật toán này dựa trên khung kĩ thuật của thuật toán **FA** của phần trước. Tuy nhiên để phân tích cho trường hợp tổng quát, các suy dẫn dựa trên tính chất đơn điệu cần phải được thay thế. Mặt khác, do sự phức tạp của hàm mục tiêu không đơn điệu, tỉ lệ xấp xỉ sẽ không đạt được là $1/10$ như thuật toán **FA**.

Để giải quyết tính không đơn điệu của hàm mục tiêu, ta phải sử dụng các phân tích không tầm thường để đưa ra một tỉ lệ xấp xỉ. Khác với trường hợp đơn điệu trong **FA**, phần này sử dụng tính chất *đơn điệu từng cặp (pairwise monotonicity)* của hàm k-submodular. Đây là một tính chất quan trọng trong phân tích lý thuyết thuật toán.

Để phù hợp với các phân tích liên quan đến hàm không đơn điệu, thuật toán sử dụng các ký hiệu bổ trợ về các tập và các bộ, quy ước như sau:

- $V_1 = \{e \in V : c(e) > B/2\}, V_2 = \{e \in V : c(e) \leq B/2\}$.
- $\mathbf{o}'_1 = \{(e, \mathbf{o}(e)) : e \in V_1\}, \mathbf{o}'_2 = \{(e, \mathbf{o}(e)) : e \in V_2\}$.
- $\mathbf{o}_1$ là lời giải tối ưu trên tập V_1.
- $\mathbf{o}_2$ là lời giải tối ưu trên tập V_2.
- (e_j, i_j) là phần tử thứ j được thêm vào vòng lặp chính của thuật toán.
- $\mathbf{x} = \{(e_1, i_1), \ldots, (e_t, i_t)\}$: k-tập $\mathbf{x}$ sau khi kết thúc vòng lặp chính, và $t =$

Algorithm 5 : **LAA**

Input: V, f, k, $B > 0$.

Output: A solution $\mathbf{s}$

$\mathbf{x} \leftarrow \mathbf{0}$; $(e_m, i_m) \leftarrow (\emptyset, 1)$; $\mathbf{x}' \leftarrow \mathbf{0}$;
for all $e \in V$ **do**
  $i_e \leftarrow \arg\max_{i \in [k]} f((e, i))$
  $(e_m, i_m) \leftarrow \arg\max_{(e', i') \in \{(e_m, i_m), (e, i_e)\}} f((e', i'))$
  if $c(e) \leq B/2$ **then**
    if $\Delta_{(e, i_e)} f(\mathbf{x}) \geq c(e) f(\mathbf{x})/B$ **then**
      $\mathbf{x} \leftarrow \mathbf{x} \sqcup (e, i_e)$
    end if
  end if
end for
$t_x = |supp(\mathbf{x})|$, $\mathbf{x}_j = \{(e_{t_x - j + 1}, i_{t_x - j + 1}), (e_{t_x - j + 2}, i_{t_x - j + 2}), \ldots, (e_{t_x}, i_{t_x})\}$, $\mathbf{x}' \leftarrow \arg\max_{\mathbf{x}_j : j \leq t_x, c(\mathbf{x}_j) \leq B} c(\mathbf{x}_j)$, // last j tuples added into $\mathbf{x}$
$\mathbf{s} \leftarrow \arg\max_{\mathbf{s} \in \{(e_m, i_m), \mathbf{x}'\}} f(\mathbf{s})$
return $\mathbf{s}_{final}$

$|supp(\mathbf{x})|$.

- $\mathbf{x}^j = \{(e_1, i_1), \ldots, (e_j, i_j)\}$: là k-tập $\mathbf{x}$ (trong vòng lặp chính) sau khi thêm j phần tử, $1 \leq j \leq t$, $\mathbf{x}^0 = \mathbf{0}$, $\mathbf{x}^t = \mathbf{x}$.

- $\mathbf{x}_j = \{(e_{t-j+1}, i_{t-j+1}), (e_{t-j+2}, i_{t-j+2}), \ldots, (e_t, i_t)\}$ là j phần tử cuối cùng được thêm vào $\mathbf{x}$.

- $\mathbf{o}_2^j = (\mathbf{o}_2 \sqcup \mathbf{x}^j) \sqcup \mathbf{x}^j$.

- $\mathbf{o}_2^{j-1/2} = (\mathbf{o}_2 \sqcup \mathbf{x}^j) \sqcup \mathbf{x}^{j-1}$.

- $\mathbf{x}^{j-1/2}$: Nếu $e_j \in supp(\mathbf{o}_2)$, thì có $\mathbf{x}^{j-1/2} = \mathbf{x}^{j-1} \sqcup (e_j, \mathbf{o}_2(e_j))$. Nếu $e_j \notin supp(\mathbf{o}_2)$, $\mathbf{x}^{j-1/2} = \mathbf{x}^{j-1}$.

- $\mathbf{u}^t = \{(u_1, i_1), (u_2, i_2), \ldots, (u_r, i_r)\}$ là tập các phần tử ở trong tập $\mathbf{o}_2^t$ mà không ở trong tập $\mathbf{x}^t$, $r = |supp(\mathbf{u}^t)|$.

- $\mathbf{u}_l^t = \mathbf{x}^t \sqcup \{(u_1, i_1), (u_2, i_2), \ldots, (u_l, i_l)\}, \forall 1 \leq l \leq r$ và $\mathbf{u}_0^t = \mathbf{x}^t$.

Giả sử rằng $\mathbf{x}'$ có T bộ cuối trong $\mathbf{x}$, có nghĩa là $\mathbf{x}' = \mathbf{x}_T$. Đặt $Q = t - T$, chúng ta có $\mathbf{x} = \mathbf{x}^Q \sqcup \mathbf{x}'$. Các bổ đề dưới đây sẽ liên kết giữa lời giải dự tuyển $\mathbf{x}$ với $\mathbf{o}_2$.

Bổ đề 2.9. $f(\mathbf{o}_2) - f(\mathbf{o}_2^j) \leq 2f(\mathbf{x}^j)$ *với mọi* $0 \leq j \leq t$.

Chứng minh. Vì f có thể không đơn điệu, với (e_j, i_j) là bộ thứ j được thêm vào lời giải dự tuyển $\mathbf{x}$ sau vòng lặp của thuật toán **LAA**. Ta phải xét 2 trường hợp dưới đây:

-Nếu $e_j \notin supp(\mathbf{o}_2)$, đặt $l \in [k]$ sao cho $l \neq i_j$ và $\mathbf{o}_l^j$ là k-tập sao cho $\mathbf{o}_l^j(e) = \mathbf{o}_2^j(e), \forall e \in V_2 \setminus \{e_j\}$ và $\mathbf{o}_l^j(e_j) = l$, ta có:

$$\begin{aligned} f(\mathbf{o}_2^{j-1}) - f(\mathbf{o}_2^j) &= f(\mathbf{o}_l^j) - f(\mathbf{o}_2^{j-1}) \\ &\quad - (f(\mathbf{o}_2^j) + f(\mathbf{o}_l^j) - 2f(\mathbf{o}_2^{j-1})) && (2.59) \\ &\leq f(\mathbf{o}_l^j) - f(\mathbf{o}_2^{j-1}) && (2.60) \\ &\leq f(\mathbf{x}_l^j) - f(\mathbf{x}^{j-1}) && (2.61) \\ &\leq f(\mathbf{x}^j) - f(\mathbf{x}^{j-1}). && (2.62) \end{aligned}$$

Với bất đẳng thức (2.60) do tính chất đơn điệu từng cặp của hàm f, bất đẳng thức (2.61) do tính k-submodular của f, và bất đẳng thức (2.62) do quy tắc chọn phần tử của thuật toán. Vậy, ta có điều phải chứng minh.

- Nếu $e_j \in supp(\mathbf{o}_2)$. Lúc này, nếu $\mathbf{o}_2^{j-1}(e_j) = i_j$, do tính chất đơn điệu từng cặp của f, tồn tại $i' \in [k]$ để $f(\mathbf{x}^{j-1} \sqcup (e_j, i')) \geq 0$. Vì vậy,

$$f(\mathbf{o}_2^j) - f(\mathbf{o}_2^{j-1}) = 0 \leq f(\mathbf{x}^j) - f(\mathbf{x}^{j-1}).$$

Nếu $\mathbf{o}_2^{j-1}(e_j) \neq i_j$, suy ra:

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

Bất đẳng thức cuối là do tính k-submodular. Tổng quát, ta có $f(\mathbf{o}_2^{j-1}) - f(\mathbf{o}_2^j) \leq 2f(\mathbf{x}^j) - 2f(\mathbf{x}^{j-1})$. Vì vậy,

$$\begin{aligned} f(\mathbf{o}_2) - f(\mathbf{o}_2^t) &= \sum_{j=1}^{t} (f(\mathbf{o}^{j-1}) - f(\mathbf{o}^j)) \\ &\leq 2\sum_{j=1}^{t} (f(\mathbf{x}^j) - f(\mathbf{x}^{j-1})) \leq 2f(\mathbf{x}^t). \end{aligned}$$

Như vậy ta cũng có điều phải chứng minh. □

Bổ đề dưới đây chỉ ra mối quan hệ giữa lời giải có được tại các bước và lời giải cuối cùng.

Bổ đề 2.10. $f(\mathbf{x}') \geq f(\mathbf{x}^t)/3$.

Chứng minh. Nhắc lại $\mathbf{x} = \mathbf{x}^t$. Nếu $c(\mathbf{x}^t) \leq B$, $\mathbf{x}' = \mathbf{x}^t$ thì bổ đề đúng. Ta xét trường hợp: $c(\mathbf{x}^t) > B$. Ta có:

$$f(\mathbf{x}^t) - f(\mathbf{x}^Q) = \sum_{j=Q+1}^{T} \Delta_{(e_j,i_j)} f(\mathbf{x}^{j-1}) \tag{2.63}$$

$$\geq \sum_{j=Q+1}^{T} c(e_j) \frac{f(\mathbf{x}^{j-1})}{B} \tag{2.64}$$

$$\geq \sum_{j=Q+1}^{T} c(e_j) \frac{f(\mathbf{x}^{Q})}{B} \tag{2.65}$$

$$\geq c(\mathbf{x}') \frac{f(\mathbf{x}^Q)}{B}. \tag{2.66}$$

Trong đó bất đẳng thức (2.64) do luật chọn bộ (e, i_j) cho vào tập $\mathbf{x}$ theo điều kiện ở dòng 6 của thuật toán **LAA**. Bất đẳng thức (2.65) là do $f(\cdot) \geq 0$.

Vì $\mathbf{x}'$ được chọn từ $\mathbf{x}$ để tổng chi phí của nó gần B nhất và mỗi phần tử $e \in supp(\mathbf{x})$ có chi phí nhiều nhất là $B/2$, nên:

$$c(\mathbf{x}') > B - \frac{B}{2} \geq \frac{B}{2}.$$

Suy ra: $f(\mathbf{x}^t) - f(\mathbf{x}^Q) \geq f(\mathbf{x}^Q)/2$. Nên $f(\mathbf{x}^Q) \leq 2f(\mathbf{x}^t)/3$. Mặt khác, do tính k-submodular của f ta có $f(\mathbf{x}^t) \leq f(\mathbf{x}^Q) + f(\mathbf{x}')$. Nên,

$$f(\mathbf{x}') \geq f(\mathbf{x}^t) - f(\mathbf{x}^Q) \geq \frac{f(\mathbf{x}^t)}{3}.$$

Ta có điều phải chứng minh. □

Bổ đề 2.11. $f(\mathbf{o}_2^t) \leq 4f(\mathbf{x}^t)$.

Chứng minh. Ta có:

$$f(\mathbf{o}_2) - f(\mathbf{x}^t) = f(\mathbf{o}_2) - f(\mathbf{o}_2^t) + f(\mathbf{o}_2^t) - f(\mathbf{x}^t) \tag{2.67}$$

$$\leq 2f(\mathbf{x}^t) + f(\mathbf{o}_2^t) - f(\mathbf{x}^t) \quad \text{(Do Bổ đề 2.9)} \tag{2.68}$$

$$\leq 2f(\mathbf{x}^t) + \sum_{e \in supp(\mathbf{o}_2^t) \setminus supp(\mathbf{x}^t)} \Delta_{(e,i_j),\forall i \in [k]} f(\mathbf{x}^t) \tag{2.69}$$

$$\leq 2f(\mathbf{x}^t) + \sum_{e \in supp(\mathbf{o}_2^t) \setminus supp(\mathbf{x}^t)} \frac{c(e) f(\mathbf{x}^t)}{B} \tag{2.70}$$

$$\leq 2f(\mathbf{x}^t) + \frac{B f(\mathbf{x}^t)}{B} = 3f(\mathbf{x}^t). \tag{2.71}$$

Trong đó bất đẳng thức (2.69) do tính k-submodular của f, bất đẳng thức (2.70) do luật lựa chọn của thuật toán. Do vậy, ta có: $f(\mathbf{o}_2^t) \leq 4f(\mathbf{x}^t)$ or $f(\mathbf{o}_2^t) \leq 4f(\mathbf{x})$. □

Từ các bổ đề trên ta suy ra bổ đề về mối quan hệ giữa lời giải cuối và lời giải tối ưu trên tập V_2 dưới đây.

Bổ đề 2.12. $f(\mathbf{x}') \geq f(\mathbf{o}_2)/18$.

Chứng minh. Áp dụng các Bổ đề 2.1 và 2.11, với $j = t$, ta có:

$$\begin{aligned} f(\mathbf{o}_2) - f(\mathbf{x}^t) &= f(\mathbf{o}_2) - f(\mathbf{o}_2^t) + f(\mathbf{o}_2^t) - f(\mathbf{x}^t) \\ &\leq 2f(\mathbf{x}^t) + f(\mathbf{o}_2^t) - f(\mathbf{x}^t) \\ &\leq 5f(\mathbf{x}^t). \end{aligned}$$

Vì vậy $f(\mathbf{x}^t) \geq f(\mathbf{o}_2)/6$. Kết hợp các suy dẫn trên với Bổ đề 2.10, ta có $f(\mathbf{x}') \geq f(\mathbf{x}^t)/3 \geq f(\mathbf{o}_2)/18$. □

Cuối cùng, các đảm bảo lý thuyết về tỉ lệ xấp xỉ và độ phức tạp truy vấn được chỉ ra qua định lý dưới đây.

Định lý 2.5. *Thuật toán* **LAA** *là thuật toán luồng 1 lần quét trả về tỉ lệ xấp xỉ là* $1/19$ *và tốn* nk *truy vấn.*

Chứng minh. Thuật toán chỉ quét một lần trên tập hợp cơ sở và mỗi phần tử e tốn k truy vấn để tìm vị trí i_e. Do đó, số lượng truy vấn là nk. Bây giờ chúng ta chứng minh tỉ lệ xấp xỉ của thuật toán. Bằng cách chọn (e_m, i_m) và $\mathbf{o}_1$ chứa nhiều nhất một phần tử nên $f(\mathbf{o}_1) \leq f((e_m, i_m))$. Từ định nghĩa của $\mathbf{o}_1', \mathbf{o}_2'$ và tính

k-submodular của f, ta suy ra:

$$f(\mathbf{o}) \leq f(\mathbf{o}'_1) + f(\mathbf{o}'_2) \tag{2.72}$$
$$\leq f(\mathbf{o}_1) + f(\mathbf{o}_2) \tag{2.73}$$
$$\leq f((e_m, i_m)) + 18f(\mathbf{x}') \leq 19f(\mathbf{s}). \tag{2.74}$$

Vậy định lý đã được chứng minh. □

2.6.3. Thuật toán tuyến tính cải tiến: RLA

Cũng tương tự trường hợp đơn điệu, ta có thể xây dựng được một thuật toán cải tiến nhằm tăng tỉ lệ xấp xỉ. Phần này giới thiệu thuật toán tuyến tính cải tiến **RLA** (*Robust Linear Approximation*). Thuật toán này cải thiện tỉ lệ xấp xỉ thành $1/5 - \epsilon$ và có độ phức tạp truy vấn là $O(kn/\epsilon)$. **RLA** giữ ý tưởng của **IFA** bằng cách sử dụng lại giải pháp của **LAA** để giới hạn phạm vi của opt và điều chỉnh ngưỡng tham lam để cải thiện tỉ lệ xấp xỉ bằng cách tiến hành $O(1/\epsilon)$ lần quét qua tập cơ sở. Chi tiết thuật toán được trình bày đầy đủ trong Thuật toán 6.

Algorithm 6 : **RLA**

Input: V, f, k, $B > 0$, $\epsilon > 0$.
Output: A solution $\mathbf{s}$

$\mathbf{s}_b \leftarrow \mathbf{LAA}(V, f, k, B)$
$\Gamma \leftarrow f(\mathbf{s}_b)$
$A \leftarrow \{(1+\epsilon)^i : i \in \mathbb{N}, \Gamma \leq (1+\epsilon)^i \leq 19\Gamma\}$
for all $e \in V$ **do**
  for all $v \in A$ **do**
    $i_v \leftarrow \arg\max_{i \in [k]} \Delta_{(e,i)} f(\mathbf{s}_v)$
    $\tau_v = 2v/(5B)$
    if $c(\mathbf{s}_v) + c(e) \leq B$ **and** $\Delta_{(e,i_v)} f(\mathbf{s}_v)/c(e)) \geq \tau_v$ **then**
      $\mathbf{s}_v \leftarrow \mathbf{s}_v \sqcup (e, i_v)$
    end if
  end for
end for
$\mathbf{s}_{final} \leftarrow \arg\max_{\mathbf{s}' \in \{\mathbf{s}_{max}, \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{|S|}\}} f(\mathbf{s}')$
return $\mathbf{s}_{final}$

Từ Định lý 2.5, ta có: $\Gamma \leq \text{opt} \leq 19\Gamma$. Vòng lặp bên ngoài cũng quét từng phần tử e trong tập cơ sở V, còn vòng lặp bên trong để xem xét từng giải pháp ứng viên $\mathbf{s}_v$ cho mỗi v được chọn lọc từ tập A. Trên cơ sở Định lý 2.5, ta xây dựng tập

A để giới hạn số nghiệm ứng viên $\mathbf{s}_v$. Định nghĩa (e, i_v) là bộ mang lại mức tăng lợi nhuận biên lớn nhất khi được thêm vào $\mathbf{s}_v$.

Khi một phần tử e đến, thuật toán sẽ xử lý các công việc sau: (1) chọn vị trí i_v với mức tăng lợi nhuận biên tối đa đối với $\mathbf{s}_v$ và e (dòng 6); (2) sử dụng ngưỡng $\tau_v = 2v/(5B)$ để thêm phần tử e vào $\mathbf{s}_v$ nếu nó có *mật độ tăng* cao mà không vi phạm ràng buộc ngân sách (dòng (8)).

Sự khác nhau của **RLA** và **IFA** nằm ở khoảng bao giá trị tối ưu và ngưỡng tham lam τ_v. Do sự chi phối của hàm f có thể không đơn điệu, nên các thiết kế của **IFA** không còn đúng trong trường hợp này. Do đó, cần thiết kế thuật toán mới này và các phân tích tỉ lệ mới. Dưới đây sẽ chỉ ra các đảm bảo lý thuyết của thuật toán **RLA**.

Ta vẫn giữ các ký hiệu $\mathbf{o}$ như một nghiệm tối ưu của bài toán trên V và giá trị tối ưu $\mathsf{opt} = f(\mathbf{o})$. Ngoài ra, luận án sử dụng thêm một số ký hiệu liên quan đến xây dựng và phân tích Thuật toán 6 như sau:

- $\mathbf{s}_v = \{(e_1, i_1), (e_2, i_2), \ldots, (e_q, i_q)\}$ là tập lời giải dự tuyển ứng với các $v \in A$ sau khi kết thúc vùng lặp ngoài.
- $\mathbf{s}_v^j = \{(e_1, i_1), (e_2, i_2), \ldots, (e_j, i_j)\}, 1 \leq j \leq q$, như vậy $\mathbf{s}_v^0 = \mathbf{0}$.
- $\mathbf{s}_v^{<e}$ là $\mathbf{s}_v$ ngay trước khi e được xem xét.
- $\mathbf{u} = \{(u_1, i_1), (u_2, i_2), \ldots, (u_r, i_r)\}$ là tập các phần tử nằm trong $\mathbf{o}$ nhưng không nằm trong $\mathbf{s}_v$, $r = |supp(\mathbf{u})|$.
- $\mathbf{u}_l = \mathbf{s}_v \sqcup \{(u_1, i_1), (u_2, i_2), \ldots, (u_l, i_l)\}, \forall 1 \leq l \leq r$ và $\mathbf{u}_0 = \mathbf{s}_v$.
- $\mathbf{o}^j = (\mathbf{o} \sqcup \mathbf{s}_v^j) \sqcup \mathbf{s}_v^j$.
- $\mathbf{o}^{j-1/2} = (\mathbf{o} \sqcup \mathbf{s}_v^j) \sqcup \mathbf{s}_v^{j-1}$.

Ta có các bổ đề và định lý dùng để phân tích đảm bảo lý thuyết dưới đây.

Bổ đề 2.13. *Với bất kỳ $v \in A$, nếu không có phần tử $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ nào để $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o})/c(o) \geq \tau_v$ và $c(\mathbf{s}_v^{<o}) + c(o) > B$, thì: $f(\mathbf{o}) \leq 3f(\mathbf{s}_v) + c(\mathbf{o})\tau_v$.*

Chứng minh. Do quy tắc lựa chọn giống nhau giữa (e, i_v) của thuật toán **RLA** và (e, i_e) của thuật toán **LAA**, chúng ta có cùng kết quả với Bổ đề 2.9, có nghĩa là, $f(\mathbf{o}) - f(\mathbf{o}^q) \leq 2f(\mathbf{s}_v)$. Vì vậy:

$$f(\mathbf{o}) - f(\mathbf{s}_v) = f(\mathbf{o}) - f(\mathbf{o}^q) + f(\mathbf{o}^q) - f(\mathbf{s}_v) \tag{2.75}$$

$$\leq 2f(\mathbf{s}_v) + \sum_{j=1}^{r}(f(\mathbf{u}_j) - f(\mathbf{u}_{j-1})) \tag{2.76}$$

$$\leq 2f(\mathbf{s}_v) + \sum_{j=1}^{r}\Delta_{(u_j,i_j)}f(\mathbf{s}_v^{<u_j}) \tag{2.77}$$

$$\leq 2f(\mathbf{s}_v) + \sum_{j=1}^{r}c(u_j)\tau_v \tag{2.78}$$

$$\leq 2f(\mathbf{s}_v) + c(\mathbf{o})\tau_v. \tag{2.79}$$

Trong đó bất đẳng thức (2.77) là do tính k-submodular, bất đẳng thức (2.78) xuất phát từ định nghĩa $\mathbf{s}_v^{<o}$, và bất đẳng thức (2.79) do định nghĩa của $\mathbf{u}$ và $\mathbf{o}$. □

Định lý 2.6. *Với $0 < \epsilon < 1/5$, thuật toán* **RLA** *trả về tỉ lệ xấp xỉ là $1/5 - \epsilon$, trong với độ phức tạp truy vấn là $O(nk/\epsilon)$.*

Chứng minh. Thuật toán cần nk truy vấn để gọi **LAA** và chỉ sử dụng 1 lần quét tập cơ sở để kết thúc vòng lặp bên ngoài (dòng 4-12). Đối với mỗi phần tử đến, chỉ cần tối đa $k \cdot \lceil \log_{(1+\epsilon)}(19) \rceil$ truy vấn để cập nhật $\mathbf{s}_v, v \in A$. Kết hợp tất cả các điều trên, số lượng truy vấn được yêu cầu nhiều nhất là:

$$\begin{aligned} nk + nk\lceil \log_{(1+\epsilon)}(19) \rceil &\leq nk + nk(1 + \log_{(1+\epsilon)}(19)) \\ &= 2nk + nk\frac{\ln(19)}{\ln(1+\epsilon)} \\ &\leq 2nk + nk\frac{\ln(19)}{\ln \frac{1}{1-\frac{\epsilon}{2}}} \\ &= 2nk - nk\frac{\ln(19)}{\ln(1-\frac{\epsilon}{2})} \\ &\leq 2nk + \frac{2}{\epsilon}nk\ln(19) = O(\frac{nk}{\epsilon}). \end{aligned}$$

Tiếp theo là tỉ lệ xấp xỉ, từ Định lý 2.5, ta có $\Gamma \leq \mathsf{opt} \leq 19.\Gamma$, nên tồn tại một số nguyên dương $v \in A$ để $\mathsf{opt}/(1+\epsilon) \leq v \leq \mathsf{opt}$. Vậy ta có:

$$f(\mathbf{s}_v^j) = \sum_{i=1}^{j}(f(\mathbf{s}_v^i) - f(\mathbf{s}_v^{i-1})) \geq \sum_{i=1}^{j}c(e_i)\theta_v = c(\mathbf{s}_v^j)\theta_v. \tag{2.80}$$

Ta xem xét các trường hợp sau:
-Trường hợp 1. Tồn tại phần tử $o \in supp(\mathbf{o}) \setminus supp(\mathbf{s}_v)$ để $\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) \geq \tau_v$ và $c(\mathbf{s}_v^{<o}) + c(o) > B$. Nhắc lại $(e_m, i_m) = \arg\max_{e \in V, i \in [k]} f((e,i))$, nên có:

$$\begin{aligned}
f(\mathbf{s}_{final}) &\geq \max\{f(\mathbf{s}_v), f((e_m, i_m))\} \\
&\geq \max\{f(\mathbf{s}_v^{<o}), f((o, \mathbf{o}(o)))\} \\
&\geq \frac{f(\mathbf{s}_v^{<o}) + f((o, \mathbf{o}(o)))}{2} \\
&\geq \frac{f(\mathbf{s}_v^{<o} \sqcup (o, \mathbf{o}(o))}{2} \\
&= \frac{\Delta_{(o,\mathbf{o}(o))} f(\mathbf{s}_v^{<o}) + f(\mathbf{s}_v^{<o})}{2} \\
&\geq \frac{\tau_v c(o) + \tau_v c(\mathbf{s}_v^o)}{2} \geq \frac{B\tau_v}{2} = \frac{v}{5} \\
&\geq \frac{\mathsf{opt}}{5(1+\epsilon)} \\
&\geq (\frac{1}{5} - \epsilon)\mathsf{opt}.
\end{aligned}$$

-Trường hợp 2. Không có phần tử o như Trường hợp 1. Từ Bổ đề 2.13, $c(\mathbf{o}) \leq B$, $\tau_v = 2v/(5B)$, và $f(\mathbf{o}) = \mathsf{opt} \geq v$, ta có:

$$\begin{aligned}
f(\mathbf{o}) &\leq 3f(\mathbf{s}_v) + B\tau_v \leq 3f(\mathbf{s}_v) + 2Bv/(5B) \\
&\leq 3f(\mathbf{s}_v) + 2\mathsf{opt}/5.
\end{aligned}$$

Suy ra: $\mathsf{opt} \leq 3f(\mathbf{s}_v) + 2\mathsf{opt}/5$, vì vậy $\mathsf{opt} \leq 5f(\mathbf{s}_v)$. Cuối cùng, $f(\mathbf{s}_{final}) \geq f(\mathbf{s}_v) \geq \mathsf{opt}/5$. Kết hợp tất cả các trường hợp trên, ta có điều phải chứng minh. □

2.7. Nghiên cứu thực nghiệm

2.7.1. Các ứng dụng dùng trong thực nghiệm

Thực nghiệm được tiến hành trên các bài toán ứng dụng của kSMK. Các bài toán ứng dụng này là các trường hợp của kSMK thể hiện cụ thể nó được áp dụng vào đâu. Đây đều là các bài toán phổ biến và có tính thực tiễn, bao gồm :

- Tối đa ảnh hưởng của k chủ đề trong giới hạn chi phí (k-topic Influence Maximization under Knapsack constraint - kIMK);

- Tối đa độ phủ thông tin của k chủ đề trong giới hạn chi phí (k-topic information Coverage Maximization under Knapsack constraint - kCMK);

- Tối ưu vị trí đặt k loại cảm biến trong giới hạn chi phí (k-type Sensor Placement under Knapsack constraint - kSPK).

2.7.1.1. Tối đa ảnh hưởng của k chủ đề trong giới hạn chi phí

Ứng dụng kIMK hoạt động theo mô hình lan truyền thông tin LT được mô tả tóm tắt dưới đây.

1. **Mô hình lan truyền LT**. Mô hình lan truyền thông tin này được gọi là mô hình Ngưỡng tuyến tính LT lần đầu tiên được xây dựng cho bài toán tối đa ảnh hưởng bởi Kempe [43], sau đó được mở rộng cho các bài toán tối đa ảnh hưởng của k chủ đề [106]. Sự lan truyền thông tin khi sử dụng mô hình này được tóm tắt như sau:

- Mạng xã hội được mô hình hóa bằng đồ thị có hướng $G = (V, E)$, trong đó V, E lần lượt đại diện cho một tập hợp người dùng và một tập hợp các liên kết. Mỗi cạnh $(u, v) \in E$ được gán các trọng số $\{w^i(u, v)\}_{i\in[k]}$, trong đó mỗi $w^i(u, v)$ biểu diễn sức ảnh hưởng của u tới v về chủ đề thứ i. Mỗi nút (node) $u \in V$ có một ngưỡng ảnh hưởng (influence threshold) đối với chủ đề i, đặt là $\theta^i(u)$, được chọn một cách ngẫu nhiên trong $[0, 1]$.

- Cho một tập hạt giống $\mathbf{s} = (S_1, S_2, \ldots, S_k) \in (k + 1)^V$, sự lan truyền thông tin chủ đề i xảy ra tại các bước rời rạc $t = 0, 1, \ldots$ như sau: Tại bước $t = 0$, tất cả các nút trong S_i được kích hoạt bởi chủ đề i. Tại bước $t \geq 1$, một nút u được kích hoạt (active) nếu $\sum_{\text{active node } v} w^i(v, u) \geq \theta^i(u)$. Quá trình phổ biến thông tin về chủ đề i kết thúc ở bước t nếu không có nút mới được kích hoạt nào và quá trình lan truyền của một chủ đề độc lập với các chủ đề khác.

- Đặt $\sigma(\mathbf{s})$ là số lượng nút được kích hoạt bởi ít nhất một trong k chủ đề sau quá trình lan truyền tập hạt giống k-tập $\mathbf{s}$, tức là,

$$\sigma(\mathbf{s}) = \mathbb{E}[|\cup_{i\in[k]} \sigma_i(S_i)|]. \tag{2.81}$$

Trong đó $\sigma_i(S_i)$ là một biến ngẫu nhiên đại diện cho tập hợp người dùng đang hoạt động cho chủ đề i với hạt giống S_i. $\sigma(\mathbf{s})$ được gọi là *ảnh hưởng của tập* $\mathbf{s}$.

2. **Bài toán kIMK**. Bài toán này được định nghĩa như sau:

Định nghĩa 2.7 (kIMK)**.** Giả sử mỗi đỉnh e có một chi phí $c(e) > 0$ cho mọi chủ đề i để thể hiện mức độ cố gắng ban đầu gây ảnh hưởng đến người dùng tương ứng đối với chủ đề đó. Với ngân sách B cho trước, bài toán yêu cầu tìm tập hạt giống $\mathbf{s}$ với $c(\mathbf{s}) = \sum_{e\in S_i, i\in k} c(e) \leq B$ sao cho $\sigma(\mathbf{s})$ là giá trị lớn nhất.

2.7.1.2. Tối đa độ phủ thông tin của k chủ đề với ràng buộc chi phí

Ứng dụng kCMK được đề xuất bởi Wang và cộng sự [143] khi xét rằng một nút không được kích hoạt (inactive node) có thể được thông báo (informed) về thông tin lan truyền bởi ít nhất một trong các hàng xóm của nó trong cùng mô hình

lan truyền của bài toán lan truyền ảnh hưởng. *Tối đa độ phủ thông tin* là tối đa số lượng mong muốn các nút được kích hoạt hoặc được thông báo. Như vậy, một nút không được kích hoạt vẫn được thông báo nếu nó có ít nhất một hàng xóm là nút kích hoạt.

Qian và cộng sự [116] đã chỉ ra rằng nếu $v \in \sigma_i(S_i)\ \forall i \in [k]$, $N(v)$ là tập các hàng xóm không được kích hoạt của v, tập các nút được kích hoạt và thông báo bởi sự lan truyền thông tin từ tập hạt giống S_i có thể là:

$$\gamma_i(S_i) = \sigma_i(S_i) \cup \left(\cup_{v \in \sigma_i(S_i)} N(v)\right). \tag{2.82}$$

Trong đó $\sigma_i(S_i)$ được cho bởi biểu thức (2.81). Như vậy, độ phủ thông tin của k chủ đề được cho bởi biểu thức (2.83) dưới đây là tổng số nút theo kì vọng được kích hoạt hoặc thông báo bởi ít nhất 1 trong số k thông tin được lan truyền:

$$\gamma(\mathbf{s}) = \mathbb{E}[|\cup_{i \in [k]} \gamma_i(S_i)|]. \tag{2.83}$$

Hàm $\gamma(\mathbf{s})$ còn gọi là hàm phủ của thông tin. Khi đó, bài toán tối đa độ phủ thông tin của k chủ đề, kCMK, được định nghĩa như sau:

Định nghĩa 2.8 (kCMK)**.** Giả sử mỗi người dùng e có một chi phí $c(e) > 0$ cho mọi i chủ đề, để thể hiện sự cố gắng bao nhiêu để bắt đầu ảnh hưởng tới người dùng tương ứng về chủ đề đó. Cho một số nguyên dương B làm ngân sách, xác suất các cạnh là $p^i_{u,v}, (u,v) \in E, i \in [k]$, bài toán cần tìm tập hạt giống $\mathbf{s}$ với tổng chi phí $c(\mathbf{s}) = \sum_{e \in S_i, i \in k} c(e) \leq B$ để $\gamma(\mathbf{s})$ là lớn nhất.

2.7.1.3. Tối ưu vị trí đặt k loại cảm biến trong giới hạn chi phí

Luận án giới thiệu chi tiết hơn về hiệu suất của các thuật toán qua bài toán tối ưu vị trí đặt k loại cảm biến trong giới hạn chi phí, kSPK, được định nghĩa:

Định nghĩa 2.9 (kSPK)**.** Cho k loại cảm biến cho các phép đo các hiện tượng khác nhau và một tập hợp V gồm n vị trí, mỗi vị trí chỉ được gán cho một cảm biến. Giả sử rằng mỗi vị trí e có chi phí $c(e) > 0$ cho mỗi loại cảm biến i. Với ngân sách $B > 0$, bài toán nhằm mục đích xác định vị trí các cảm biến này để tối đa hóa thông tin thu được với tổng chi phí nhiều nhất là B.

Đặt X^i_e là một biến ngẫu nhiên đại diện cho quan sát được thu thập từ cảm biến loại i và thông tin thu được của $\mathbf{s}$ là

$$f(\mathbf{s}) = H(\cup_{e \in supp(\mathbf{s})} \{X^i_e\}).$$

Trong đó, H là một hàm Entropy [109].

2.7.2. Thực nghiệm cho trường hợp hàm mục tiêu đơn điệu

Để đánh giá các thuật toán được đề xuất, phần thực nghiệm so sánh hiệu suất giữa các thuật toán trình bày ở trên với các thuật toán tiên tiến hiện nay cho bài toán kSMK được liệt kê bên dưới:

- **Greedy**: Thuật toán tham lam cho tỉ lệ $(1/2 - 1/(2e))$ trong $O(n^4k^3)$ thời gian được đề xuất trong [134].
- **DS**[1]: Thuật toán luồng tất định của [115] đã trả về tỉ lệ xấp xỉ là $1/4 - \epsilon$, cần 1 lần quét và $O(kn\log(n)/\epsilon)$ truy vấn.
- **RS**: Một thuật toán luồng khác trong [115] trả lại tỉ lệ xấp xỉ là $k/(4k-1) - \epsilon$ theo kỳ vọng, trong 1 lần quét và $O(kn\log(n)/\epsilon)$ truy vấn.

2.7.2.1. Thiết lập cho thực nghiệm

a) Bộ dữ liệu

Thực nghiệm thu thập ba phép đo chính: giá trị ước lượng của hàm mục tiêu, số lượng truy vấn và thời gian chạy.

Tập dữ liệu được sử dụng là các tập phổ biến trong các công bố về tối ưu hàm submodular [115, 106, 105, 106] để minh họa hiệu suất của các thuật toán được so sánh (Bảng 2.5). Các dữ liệu Facebook, Hept, Enron được sử dụng cho 2 ứng dụng là kIMK và kCMK. Các dữ liệu được tổ chức dưới dạng đồ thị với các đỉnh được đánh số, các cạnh biểu thị liên kết giữa các đỉnh. Chẳng hạn Facebook [93] là một đồ thị con của dữ liệu người dùng Facebook, một cạnh biểu thị tương tác giữa hai người dùng, loại đồ thị là có hướng.

Intel Lab sensors [15] và DF-AMS-WSN [72] là 2 bộ dữ liệu cảm biến được dùng cho ứng dụng kSPK, trong đó các phần tử là các cảm biến được đánh số tăng dần. Dữ liệu cảm biến Intel Lab [15] được thu thập từ 2.3 triệu bản ghi đọc được của hệ thống dự báo thời tiết Mica2dot gồm về nhiệt độ, hơi nước, ánh sáng và điện thế. DF-AMS WSN [72] là bộ dữ liệu được thu thập từ đầu ra của phần mềm mô phỏng NS-2 một hệ thống mạng cảm biến không dây, thu thập các thông số về nhiệt độ, hơi nước, tốc độ gió, và năng lượng tiêu thụ. Dữ liệu đã được tiền xử lý để loại bỏ các trường dữ liệu bị mất và phù hợp với phần đọc dữ liệu đầu vào của thực nghiệm. Do vậy, khác với các tập dữ liệu Facebook, Hept và Enron, các tập dữ liệu cảm biến không có thông tin cạnh (Bảng 2.5).

Kết quả của các thuật toán thông qua ba phép đo trên được biểu thị thông qua các hình vẽ được đánh số dưới đây, trong đó các thuật ngữ Fig, K và M lần lượt là

[1]Bài toán kSMK là trường hợp đặc biệt của tối đa k-submodular dưới ràng buộc chi phí [115] với $\beta = 1$.

viết tắt của thuật ngữ Hình, hàng nghìn và hàng triệu.

Bảng 2.5: Tập dữ liệu

Tập dữ liệu	**Số đỉnh**	**Số cạnh**	**Loại**	**Trường hợp**
Facebook [93]	4039	88234	Có hướng	kIMK, kCMK
Hept [32]	15233	58894	Có hướng	kIMK, kCMK
Enron [77]	36692	367662	Có hướng	kIMK, kCMK
Intel Lab sensors[15]	56	Không có	-	kSPK
DF-AMS WSN[72]	100	Không có	-	kSPK

Các thực nghiệm được chạy trên hệ điều hành Linux với cấu hình 2 $\times$ Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz và 4 $\times$ 16 GB DIMM ECC DDR4 @ 2400MHz.

b) Các thiết lập thông số

Trong phần thực nghiệm, các tham số k, ϵ, B, chi phí từng đỉnh, các tham số của ứng dụng cần dùng... được thiết lập đối với các thuật toán là như nhau. Thiết lập này để đảm bảo tính công bằng giữa các thuật toán.

1. **Đối với kIMK và kCMK.** Trong thử nghiệm, 2 bài toán trên đã chạy trên ba cơ sở dữ liệu khác nhau: Facebook, Hept và Enron, những cơ sở dữ liệu phổ biến trong các bài toán lan truyền thông tin [105, 71, 115] và thiết lập mô hình như trong nghiên cứu của [106] để sử dụng cách tính toán hàm mục tiêu do các tác giả cung cấp.

Vì tính toán $\sigma(\cdot)$ là #P-khó [33], chúng ta sử dụng phương pháp lấy mẫu trong [106, 16] để có một ước lượng $\hat{\sigma}(\cdot)$ với một tỉ lệ xấp xỉ là (λ, δ):

$$\Pr[(1+\lambda)\sigma(\mathbf{s}) \geq \hat{\sigma}(\mathbf{s}) \geq (1-\lambda)\sigma(\mathbf{s})] \geq 1-\delta. \tag{2.84}$$

Trong thử nghiệm, các tham số được thiết lập như sau: $\lambda = 0.8, \delta = 0.2$, $k = 3$ và $\epsilon = 0.1$ như trong [106]. Hàm ước lượng $\hat{\gamma}(\cdot)$ của $\gamma(\cdot)$ được tính toán thông qua các biểu thức (2.83) và (2.84). Các mốc ngân sách B của kIMK, kCMK được thiết lập với một vài mốc từ 0.5K đến 2K, minh họa thực tế rằng chi phí để tác động đến k chủ đề thông qua các mạng cỡ lớn như mạng xã hội, không phải là một con số nhỏ. Chi phí cho từng phần tử gán theo mô hình LT [115] chuẩn. Theo đó, giá trị chi phí từ 1 đến 10 với Facebook và từ 1 đến 50 với Hept và Enron.

Theo các phân tích của [143, 116], bài toán kCMK là một biến thể của kIMK. Xác suất cạnh của mỗi cạnh $(u, v) \in E$ có vec-tơ xác suất là $(1/k, \ldots, 1/k)$ giống như thiết lập của Qian [116]. Bên cạnh đó, ta sẽ giữ nguyên các thiết lập cho cả

kIMK và kCMK vì khi chạy các bài toán này, sử dụng chung số lần lấy mẫu trong cùng một lần thực nghiệm. Các kết quả của kIMK và kCMK được thu thập gồm các giá trị hàm mục tiêu $\hat{\sigma}, \hat{\gamma}$, thời gian chạy và số lượng truy vấn được đo bằng cách đếm số lần gọi hàm mục tiêu.

2. **Đối với kSPK.** Ta sử dụng 2 tập dữ liệu Intel Lab [15] và DF-AMS WSN [72] để minh họa bài toán kSPK.

Thiết lập các tham số và phương pháp tính toán hàm mục tiêu được thực hiện tương tự nghiên cứu [106] để sử dụng phương pháp tính toán hàm mục tiêu của các tác giả.

Ngoài ra, thực nghiệm đặt $k = 3$, $\epsilon = 0.1$, phạm vi chi phí các đỉnh là từ 1 đến 10 như trong thử nghiệm của kIMK, nhưng các giá trị của B gồm một số mốc từ 10 đến 50. Thiết lập này là do số lượng cảm biến và để có sự tương đồng giữa các thuật toán. Đối với ứng dụng này, thời gian, giá trị hàm mục tiêu, và số lượng truy vấn cũng được đưa ra. Trong đó, số lượng truy vấn cũng là đếm số lời gọi hàm mục tiêu.

2.7.2.2. Nhận xét kết quả thực nghiệm

Để cung cấp một thử nghiệm toàn diện, các thuật toán trên đã được chạy độc lập nhiều lần và thu thập kết quả về giá trị hàm mục tiêu, số lượng truy vấn và thời gian chạy theo các mốc B. Đối với mỗi cột mốc, các giá trị trung bình đã được tính toán. Bảng 2.6 minh hoạ một trường hợp, thu thập các kết quả khi chạy ứng dụng kIMK và kCMK trên bộ dữ liệu Enron với hàm mục tiêu là các hàm đơn điệu. Cụ thể hơn, Hình (2.1)-(2.3) minh họa kết quả trên các bộ dữ liệu đã mô tả ở trên cho ba trường hợp kIMK, kCMK và kSPK.

Thực nghiệm thể hiện sự "đánh đổi" giữa chất lượng giải pháp và số lượng truy vấn của thuật toán với các giá trị khác nhau của ngân sách B. Trong các trường hợp này, xu hướng chung **IFA+** cho kết quả tốt nhất, theo sau là nhóm các thuật toán **IFA**, DS và RS, còn **FA** thì cho kết quả dao động. Greedy là thuật toán cho lượng truy vấn tốn kém nhất, thời gian chạy lâu nhất và thực nghiệm phải ràng buộc thời gian buộc phải dừng thuật toán Greedy. Sự "đánh đổi" thể hiện ở chỗ, để tăng cường chất lượng lời giải, **IFA+** đã phải tốn thêm truy vấn so với **IFA** và **FA**, tuy nhiên số lượng truy vấn bỏ ra là chấp nhận được nên thời gian chạy vẫn đảm bảo.

Điển hình, có những thực nghiệm **IFA+** và **IFA** cho chất lượng lời giải *cao hơn 1.5 lần* so với DS và RS trong khi số lượng truy vấn có thể *thấp hơn tới 20 lần* so với 2 thuật toán này. Đặc biệt, so với thuật toán Greedy, các thuật toán tiết kiệm vài trăm lần số truy vấn mà vẫn đảm bảo chất lượng.

Thuật toán	Ngân sách B	Hàm ảnh hưởng $\sigma(\cdot)$	Hàm phủ $\gamma(\cdot)$	Số truy vấn	Thời gian (s)
FA					
	0.5K	3926.04	21524.90	146769	62844.90
	0.7K	3889.35	23570.80	146769	62707.20
	1K	5237.78	25684.00	146769	62537.00
	1.2K	4458.08	26696.10	146769	62619.70
	1.5K	4797.48	27869.40	146769	62695.80
	2K	3201.38	29695.90	146769	63130.90
IFA					
	0.5K	4237.93	23536.20	1027377	4404160.00
	0.7K	3962.74	25353.10	1027377	444790.00
	1K	5503.80	27513.50	1027377	453721.00
	1.2K	4742.44	28764.50	1027377	451752.00
	1.5K	4393.87	30001.50	1027377	449386.00
	2K	5531.32	31557.90	1027377	447500.00
IFA+					
	0.5K	4157.21	25669.00	2220113	1020905.55
	0.7K	4370.86	28031.80	3247370	1451624.34
	1K	5403.75	30373.9	3496834	1562630.05
	1.2K	5237.63	31828.90	2758080	1260321.25
	1.5K	5422.81	33249.20	3393604	1531813.80
	2K	5692.27	35380.80	3572544	5395415.93
DS					
	0.5K	3265.59	19318.40	1462221	687488.00
	0.7K	2816.11	21618.10	1737756	803329.00
	1K	2981.22	23729.10	1164306	858652.00
	1.2K	3384.84	24380.20	1857588	844945.00
	1.5K	3256.41	25340.50	19237297	894806.00
	2K	3522.43	28394.40	2015664	936511.00
RS					
	0.5K	3586.64	22973.40	2523867	1169420.00
	0.7K	3338.97	25541.70	3370704	1583520.01
	1K	3421.53	28232.70	3645741	1720209.99
	1.2K	3228.90	29098.50	4206699	2029769.99
	1.5K	3256.41	29954.40	1923729	2138868.05
	2K	4026.95	31273.00	4696509	2253830.00
Greedy					
	0.5K	1183.32	16317.61	5011576	2584789.99
	0.7K	1210.84	16317.61	5014462	2584780.01
	1K	1201.66	16317.61	5038819	2587930.01
	1.2K	1201.66	16317.61	5037405	2587940.00
	1.5K	1357.60	16317.61	5070441	2588510.00
	2K	1201.66	16317.61	5075261	2588420.00

Bảng 2.6: Các kết quả thu thập được từ các thuật toán cho ứng dụng kIMK và kCMK trên bộ dữ liệu Enron, trường hợp đơn điệu.

Các phân tích cụ thể được chỉ ra trên từng ứng dụng như sau:

1. **Đối với kIMK và kCMK.** Các đường vẽ thuật toán Greedy của [134] không phải là kết quả cuối cùng vì mất quá nhiều thời gian để hoàn thành việc chạy thuật toán này. Để thực nghiệm phù hợp với cấu hình hệ thống, tác giả phải giới hạn thời gian xử lý Greedy tùy theo từng bộ dữ liệu, trong đó $4K$ nút của Facebook chạy trong vòng 12 giờ, $15K$ nút của Hept chạy trong vòng 2 ngày và $36K$ nút của Enron đã làm việc trong 4 ngày.

Hình 2.1 thể hiện chất lượng của thuật toán thông qua giá trị của các hàm mục tiêu $\sigma(\cdot)$ và $\gamma(\cdot)$. Đường vẽ của Greedy chỉ hiển thị ước tính của $\sigma(\cdot)$ và $\gamma(\cdot)$ trong thời gian bị giới hạn. Trong phần còn lại, **IFA+** luôn mang lại kết quả tốt nhất, tiếp theo là **IFA**, DS và RS. Giá trị của **FA** dao động vì các đường của **FA** đánh dấu điểm thấp nhất trong một số trường hợp và tốt hơn trong các trường hợp khác.

Trong Hình 2.1, khoảng cách giữa **FA**, **IFA**, **IFA+**, DS và RS trở nên lớn hơn khi $B \geq 1.5K$. Và khi B và n (kích thước của V) tăng lên, hiệu suất của **IFA+** và **IFA** có vẻ tốt hơn. Chẳng hạn, ở mức $B = 2K$, với Hept ($n \approx 15K$ nút) **IFA+** và **IFA** cao hơn khoảng 1.5 lần tương ứng so với DS và RS, trong khi với Enron ($n \approx 36K$ nút), **IFA+** và **IFA** cao hơn khoảng 1.25 lần so với cả DS và RS. Đặc biệt, **IFA+** và **IFA** cho thấy kết quả vượt trội trong mạng Enron.

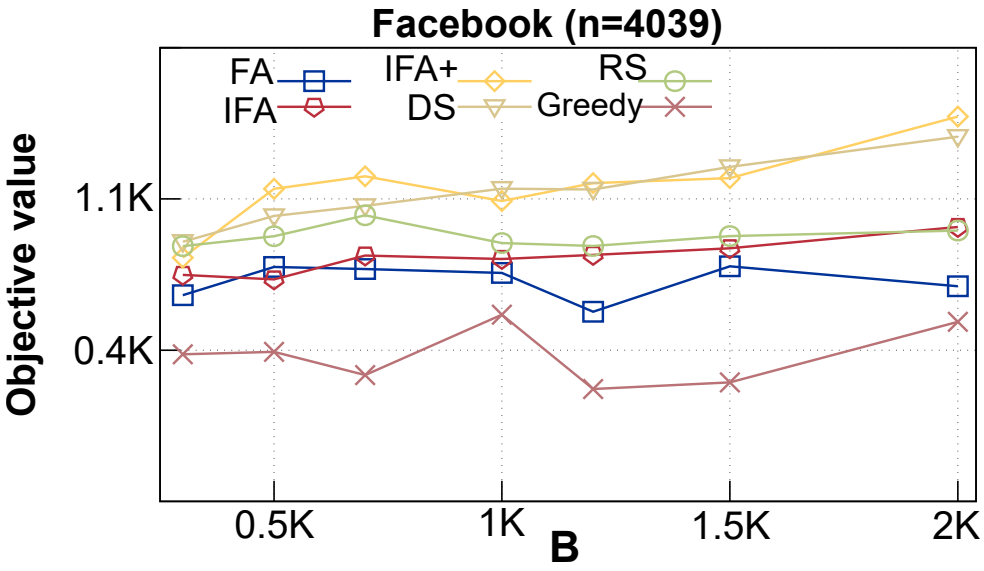
Đối với kCMK, kết quả các trường hợp (Hình 2.1 d, e, và f) có xu hướng giống như của kIMK (Hình 2.1 a, b, và c). Có thể thấy rằng hiệu suất về chất lượng giải pháp của **IFA+** luôn cao hơn các phương pháp khác. **FA**, **IFA** và DS dường như không khác nhau lắm, trong khi RS có chút dao động.

Nói chung, kết quả sẽ phân cụm thuật toán thành ba nhóm từ trên xuống dưới: **IFA+**-DS, **IFA**-**FA**-RS và Greedy. Chúng ta có thể thấy khoảng cách giữa các thuật toán trong mỗi nhóm khi B tăng từ $0.7K$ lên $1.5K$ là tương đối nhỏ và tách biệt khi B tăng hơn $1.5K$. Các đường này cũng dao động theo từng mốc B, tuy nhiên ta có thể nhận thấy xu hướng chung là **IFA+**, **IFA**, DS và RS sẽ tăng lên trong khi các giá trị **FA** dường như không thể đoán trước.

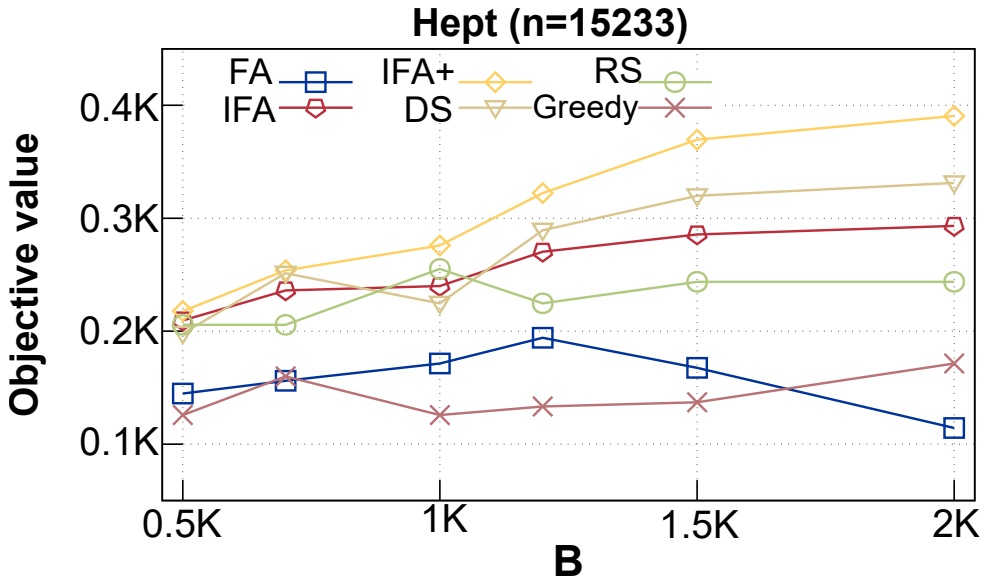
Những kết quả này phản ánh sự đảm bảo về mặt lý thuyết khi tỉ lệ xấp xỉ của **IFA+** tốt hơn các tỉ lệ khác, **FA** là thấp nhất và vai trò của **FA** chỉ là giới hạn phạm vi của giá trị tối ưu.

Thứ hai, Hình 2.2 hiển thị số lượng truy vấn được gọi và thời gian cần thiết để chạy các thuật toán này. Như đã đề cập, tác giả đã chạy kIMK và kCMK trong cùng một thử nghiệm. Do đó, kết quả của các truy vấn và thời gian chạy thuộc về cả kIMK và kCMK.

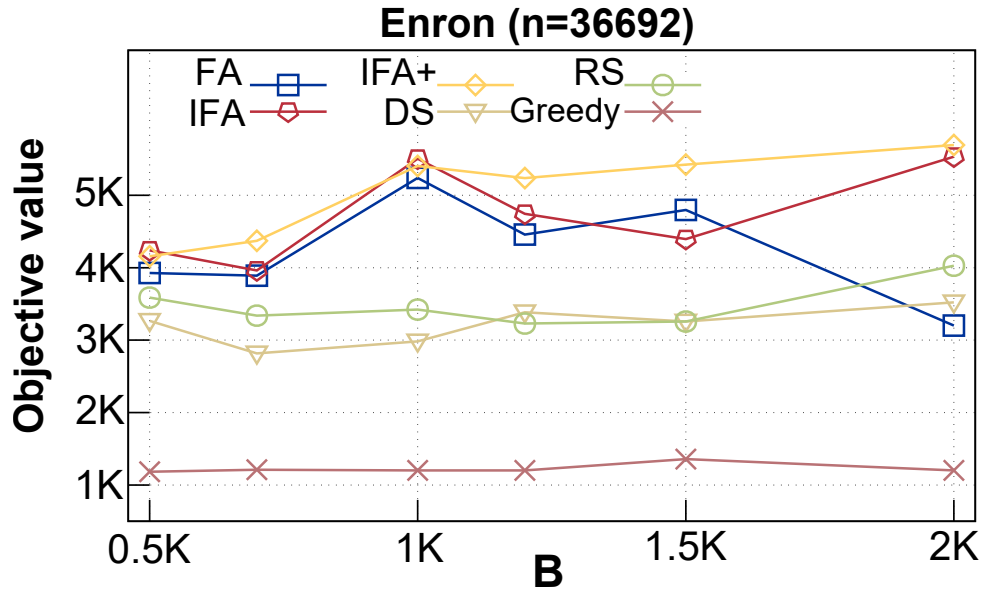
Thuật toán Greedy chiếm số lượng truy vấn nhiều nhất trong thử nghiệm này. Mặc dù Greedy được xử lý trong thời gian giới hạn, nhưng nó vẫn tiêu tốn gần


Facebook (n=4039)
FA
IFA
IFA+
DS
RS
Greedy
Objective value
1.1K
0.4K
0.5K
1K
1.5K
2K
B

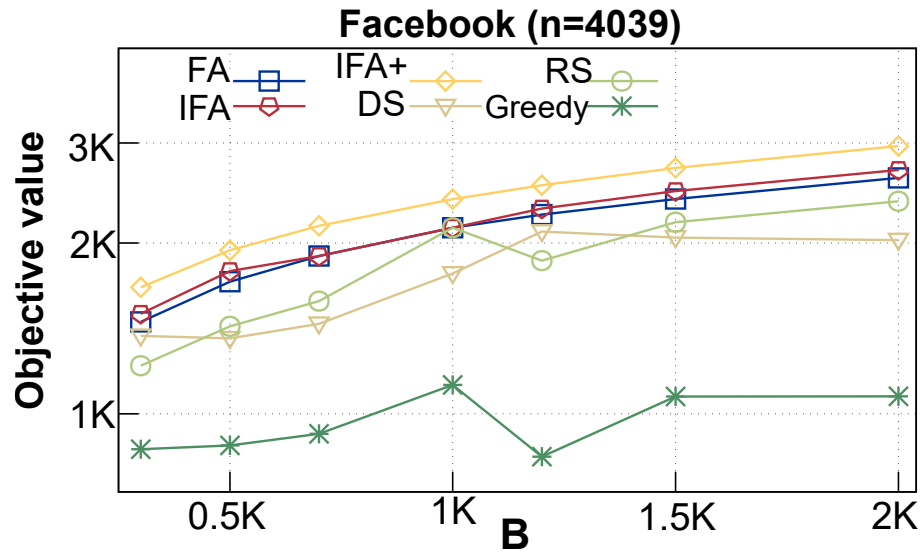
(a)


Hept (n=15233)
FA
IFA
IFA+
DS
RS
Greedy
Objective value
0.4K
0.3K
0.2K
0.1K
0.5K
1K
1.5K
2K
B

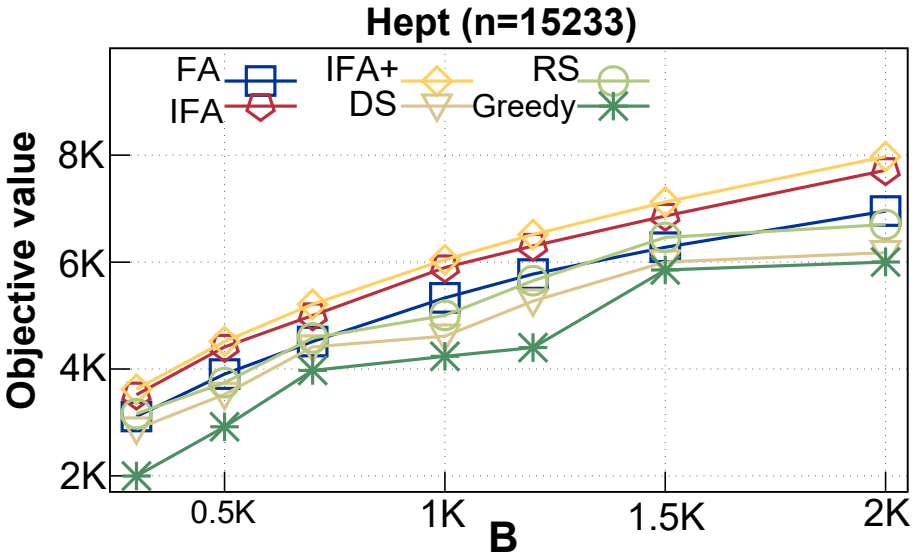
(b)


Enron (n=36692)
FA
IFA
IFA+
DS
RS
Greedy
Objective value
5K
4K
3K
2K
1K
0.5K
1K
1.5K
2K
B

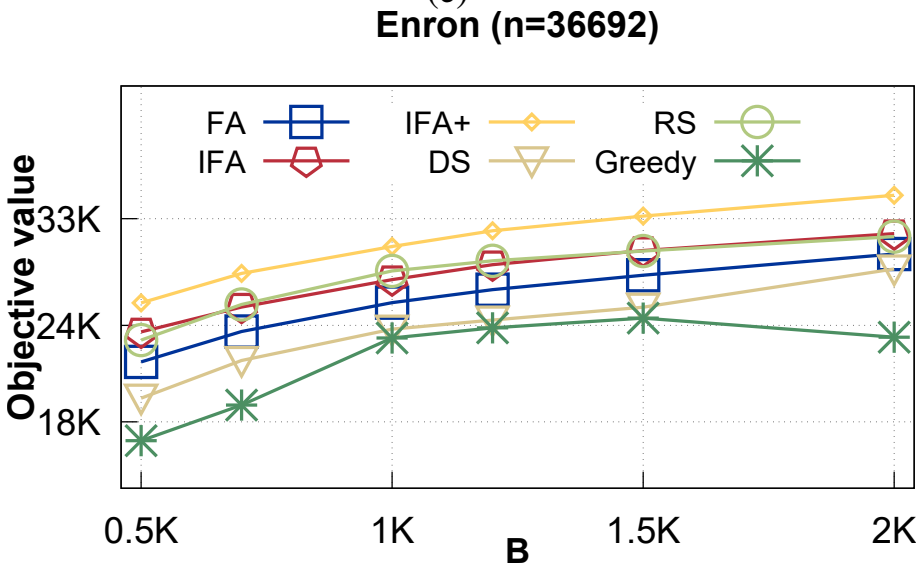
(c)



(d)

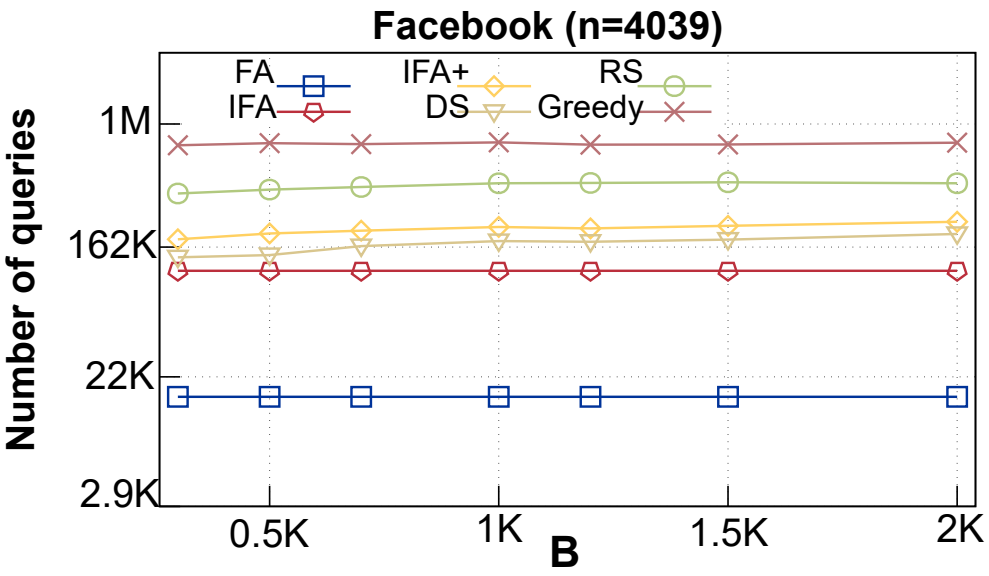


(e)

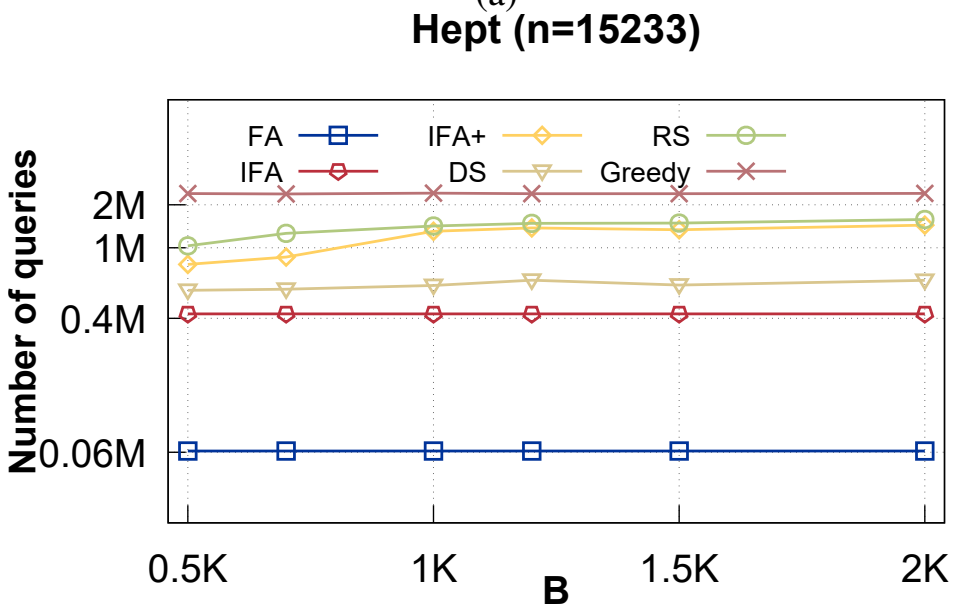


(f)

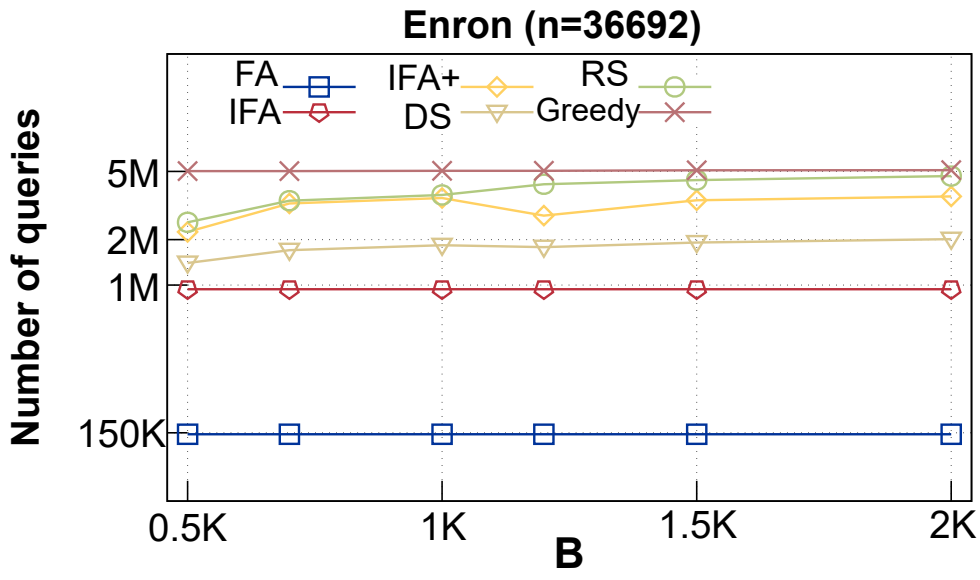
Hình 2.1: Chất lượng lời giải của các thuật toán trong kIMK (Hình a, b và c) và kCMK (Hình d, e và f).


Facebook (n=4039)
FA
IFA
IFA+
DS
RS
Greedy
Number of queries
1M
162K
22K
2.9K
0.5K
1K
1.5K
2K
B

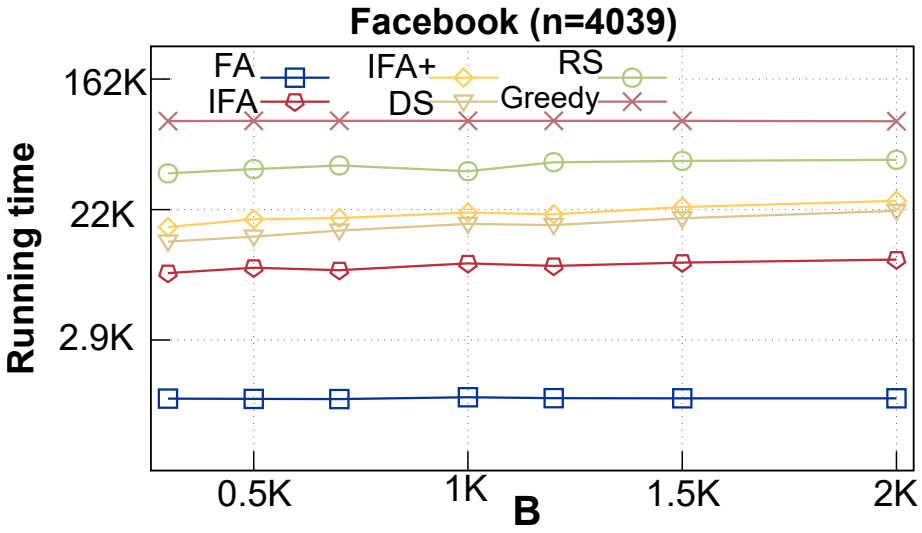
(a)


Hept (n=15233)
FA
IFA
IFA+
DS
RS
Greedy
Number of queries
2M
1M
0.4M
0.06M
0.5K
1K
1.5K
2K
B

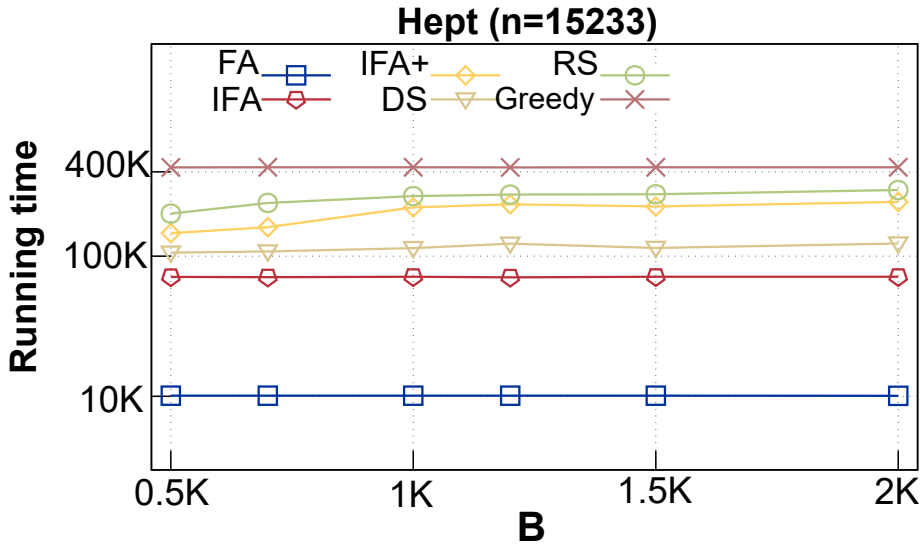
(b)


Enron (n=36692)
FA
IFA
IFA+
DS
RS
Greedy
Number of queries
5M
2M
1M
150K
0.5K
1K
1.5K
2K
B

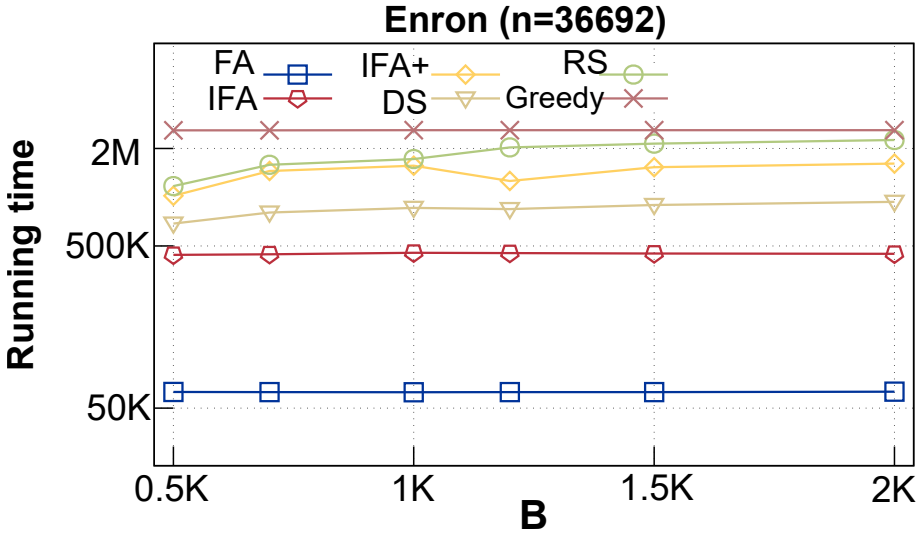
(c)



(d)



(e)



(f)

Hình 2.2: Số lượng truy vấn (a, b, và c) và thời gian chạy (d, e và f) của các thuật toán trên kIMK và kCMK.

$1M$-$5M$ truy vấn, cao hơn nhiều các truy vấn khác. Bây giờ ta tập trung vào kết quả của phần còn lại.

FA cho thấy một lợi thế so với những thuật toán khác về độ phức tạp truy vấn, nó thấp hơn hẳn từ vài đến hàng chục lần so với các loại còn lại. Tuy nhiên, như trên, nó cho chất lượng lời giải thấp. Phân tích kĩ hơn, RS cao hơn khoảng 15-20 lần và 4-5 lần, trong khi DS cao hơn khoảng $7 - 10$ lần và 1.5 lần lần lượt so với **FA** và **IFA**. Bên cạnh đó, số lượng truy vấn của **IFA+** luôn cao hơn DS và thấp hơn RS. Đáng chú ý, các đường của **IFA+**, **IFA** và **FA** tất định và đi thẳng qua các mốc B. Nhìn chung, số lượng truy vấn của RS là cao nhất, tiếp theo là **IFA+** và DS cao hơn một chút so với phần còn lại. Như vậy, số lượng truy vấn của các thuật toán trong luận án cho kết quả tốt hơn các thuật toán khác. Lý giải sự tốt hơn thể hiện ở chỗ, nó vừa đảm bảo chất lượng lời giải, vừa đảm bảo không lớn quá cách biệt với các thuật toán DS và RS là các thuật toán tiên tiến. Bên cạnh đó, các đường truy vấn gần như là nằm ngang, nên các thuật toán này đều có xu hướng ổn định với mọi V và B.

Vì độ phức tạp của truy vấn ảnh hưởng trực tiếp đến thời gian chạy, nên xu hướng của biểu đồ thời gian tương đồng với xu hướng của biểu đồ truy vấn trong đó đường **FA** được vẽ thường thấp nhất. Nó cho thấy thời gian chạy của **FA** nhanh hơn vài lần đến hàng chục lần so với những thuật toán khác. **IFA** chạy nhanh hơn đáng kể so với DS và RS, trong khi Greedy chạy chậm nhất. **IFA+** nhanh hơn RS nhưng chậm hơn DS.

Từ các số liệu trên, có thể thấy sự đánh đổi giữa chất lượng giải pháp của các thuật toán được đề xuất và độ phức tạp của truy vấn. **FA** cố gắng nhắm đến giá trị gần tối ưu bằng cách chia tập cơ sở thành hai tập con theo giá trị chi phí của các phần tử và giảm độ phức tạp của truy vấn bằng điều kiện lọc của Thuật toán 2. Do đó, độ phức tạp của truy vấn giảm đáng kể. Tuy nhiên, hiệu suất của **FA** về chất lượng giải pháp không cao. **IFA** và **IFA+** nâng cao **FA** bằng cách sử dụng **FA** làm đầu vào và ngưỡng tham lam giảm dần. Do đó, giá trị hàm mục tiêu của **IFA** tốt hơn **FA** trong khi số lượng truy vấn cao hơn một chút nhưng vẫn là tất định. **IFA+** xây dựng lại giải pháp trong giai đoạn thứ hai để nâng cấp hiệu suất. Nó dẫn đến giá trị mục tiêu tăng lên, nhưng số lượng truy vấn cũng tăng lên. Hơn nữa, khi tập hợp cơ sở và giá trị B tăng lên, chất lượng giải pháp sẽ cải thiện trong khi thời gian chạy và độ phức tạp của truy vấn là tuyến tính. Điều này cực kỳ quan trọng khi làm việc với dữ liệu lớn.

Bên cạnh đó, sự khác biệt về chất lượng giữa thuật toán của luận án và thuật toán trong [115] là do cách xây dựng thuật toán. DS và RS tập trung vào việc tìm kiếm nhiều giải pháp khả thi hơn **FA** và **IFA**; do đó, họ có nhiều cơ hội hơn để

tìm ra giải pháp tốt hơn. Trong **IFA+**, việc chọn lại một phần nhỏ các phần tử trong giải pháp và thay thế nó bằng các phần tử tốt hơn trong tập hợp V sẽ tăng hiệu suất với một chút chi phí truy vấn cao hơn. Do đó, khi dữ liệu đầu vào tăng lên, các thuật toán của luận án vượt qua các thuật toán khác, cả về chất lượng giải pháp và số lượng truy vấn.

Ngoài ra, từ các số liệu trên, thử nghiệm cho thấy các thuật toán của tác giả vượt trội hơn Greedy vài chục lần về số lượng truy vấn và thời gian chạy. Điều này phản ánh chính xác lý thuyết đảm bảo rằng độ phức tạp truy vấn của các thuật toán được đề xuất là $O(kn)$ trong khi độ phức tạp của thuật toán Greedy là $O(n^4k^3)$.

2. **Đối với kSPK.** Kết quả được thể hiện ở Hình 2.3. Đầu tiên, liên quan đến hàm mục tiêu của kSPK, đó là một ước lượng của hàm Entropy. Bên cạnh đó, số lượng nút cảm biến trong trường hợp này là ít. Do đó sự phân biệt giữa các giá trị hàm mục tiêu của các thuật toán thực nghiệm là không lớn.

Tuy nhiên, chúng ta có thể thấy kết quả của **IFA+**, RS, và Greedy tốt hơn các kết quả khác. Đặc biệt, đường **IFA+** luôn nằm trên các đường khác khi $B \leq 40$. **FA** và **IFA** trùng nhau, trong khi kết quả DS dao động nhẹ. Nhìn chung, xu hướng chung của các đường là tăng.

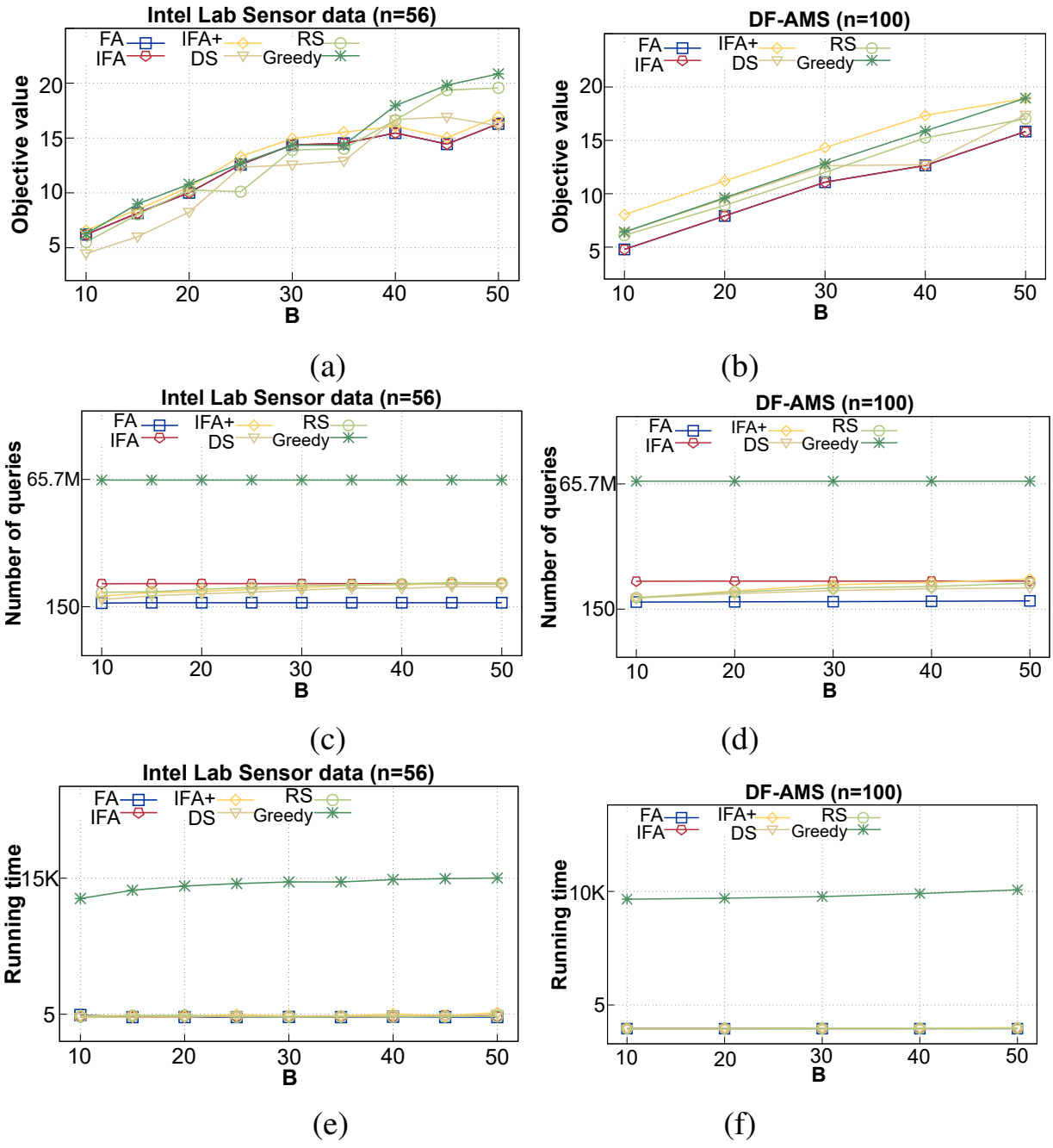
Thứ hai, khoảng cách giữa đường Greedy và các đường khác trong Hình 2.3(c)-(f) là rất lớn. Trong khi Greedy cần hàng triệu truy vấn để đưa ra giải pháp cuối cùng, các thuật toán còn lại chỉ sử dụng khoảng 150 truy vấn. Tương tự như vậy, thời gian chạy của Greedy cũng gấp hàng nghìn lần những cái khác. Hơn nữa, các đường truy vấn và đường thời gian của các thuật toán trên gần như nằm ngang so với các mốc của B. Mặt khác, số lượng truy vấn và thời gian của **FA** là nhỏ nhất và có một chút khác biệt giữa **IFA**, **IFA+**, DS và RS. Kết quả này minh họa độ phức tạp truy vấn của các thuật toán của tác giả được tốt hơn các thuật toán khác.

Tóm lại, từ ba ứng dụng kIMK, kCMK và kSPK, các thuật toán được đề xuất trong luận án được thực nghiệm là hoạt động tốt hơn hoặc có hiệu quả tương đương với các thuật toán hiện đại nhất.

2.7.3. Thực nghiệm cho trường hợp hàm mục tiêu không đơn điệu

2.7.3.1. Thiết lập cho thực nghiệm

Để thực nghiệm cho trường hợp kSMK không đơn điệu, luận án tiến hành so sánh thuật toán **LAA** và **RLA** với 2 thuật toán DS, RS [115] với 3 tập dữ liệu là Facebook [93], Hept [32] và Enron [77] cho bài toán kIMK. Các cấu hình, các thiết lập tham số $k = 3, \lambda = 0.8, \delta = 0.2, \epsilon = 0.1$ giống như phần thực nghiệm với kIMK đơn điệu. Chi phí các đỉnh lấy ngẫu nhiên từ 1 đến 50, các mốc B chạy từ $0.5K$ đến $2K$ đối với tất cả các bộ dữ liệu. Các thuật toán so sánh được nêu trong



Hình 2.3: Hiệu quả của các thuật toán thông qua bài toán kSPK: (a), (b) Thông tin thu được; (c), (d) số lượng truy vấn; (e), (f) thời gian chạy.

Bảng 2.4. Cũng như trường hợp với hàm mục tiêu đơn điệu, các tham số được thiết lập là đồng nhất, công bằng cho tất cả các thuật toán được so sánh.

Phần thực nghiệm được tiến hành đo để lấy các giá trị của hàm mục tiêu, số lượng câu truy vấn và thời gian chạy. Vì số lượng truy vấn gọi hàm mục tiêu chiếm phần lớn thời gian chạy của các thuật toán, nên xu hướng của các đường truy vấn sẽ đại diện cho thời gian chạy của các thuật toán. Bảng 2.7 minh họa kết quả đo được khi chạy các thuật toán với bộ dữ liệu Enron.

Các kết quả thực nghiệm đối với hàm mục tiêu và số truy vấn được trình bày

Thuật toán	Ngân sách B	Hàm ảnh hưởng $\sigma(\cdot)$	Số truy vấn	Thời gian (s)
LAA				
	0.5K	6472.01	220153	3553.38
	1K	9171.1	220153	3584.29
	1.5K	6472.01	220153	3575.74
	2K	6472.01	220153	3582.33
RLA				
	0.5K	8448.371	2773138	44443.9
	1K	12251.3	2989093	48239.1
	1.5K	14564.8	3202633	52022.8
	2K	15700	3105268	50774.6
RS				
	0.5K	7404.96	2816222	48111.4
	1K	11618.5	3091001	51221.4
	1.5K	13084.3	3400023	52920.8
	2K	13377.9	3405291	53228.1
DS				
	0.5K	6638.04	2173562	39549.3
	1K	10951.1	3091001	51221.4
	1.5K	13084.3	2315041	42240.9
	2K	16009.4	2951691	47779.2

Bảng 2.7: Kết quả chạy các thuật toán cho ứng dụng kIMK trên bộ dữ liệu Enron, trường hợp không đơn điệu.

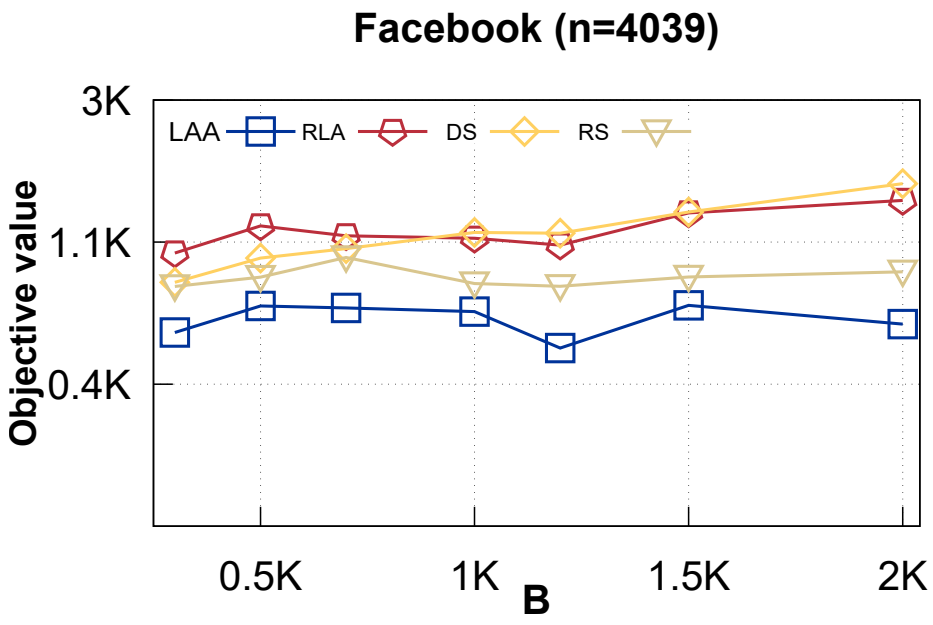
trong các Hình 2.4 - 2.6 .

2.7.3.2. Nhận xét kết quả thực nghiệm

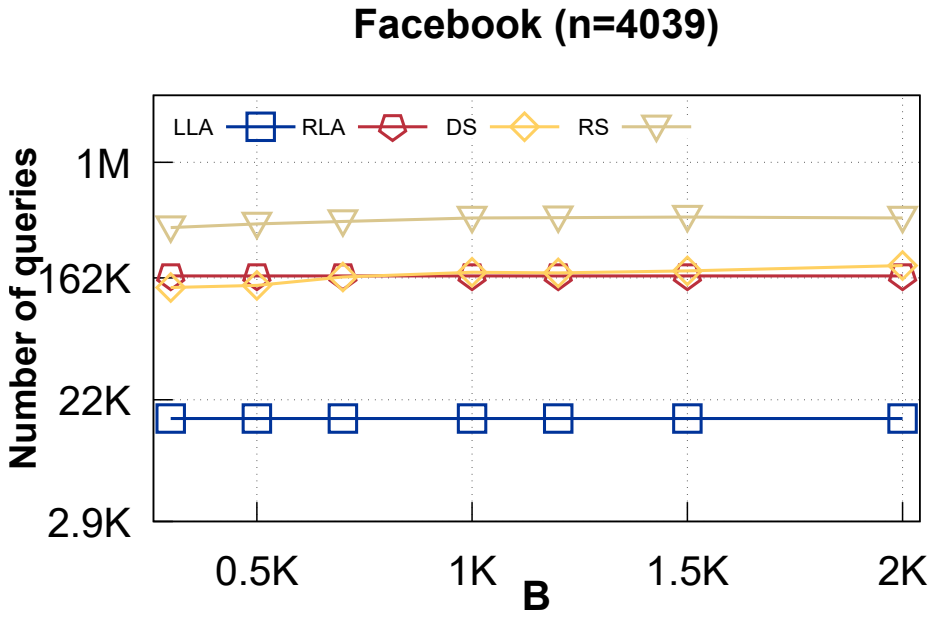
Các kết quả thực nghiệm cho thấy, các thuật toán của luận án cho chất lượng lời giải tương đương với DS và RS với số lượng truy vấn có thể tương đương, hoặc cao hơn một chút so với DS nhưng thấp hơn vài lần so với RS.

Đầu tiên, Hình 2.4(a)-2.6(a) thể hiện hiệu suất của thuật toán thông qua các giá trị của $\sigma(\cdot)$. Đối với Facebook, $\mathbf{RLA}$ tương đương với DS, theo sau là RS, trong khi đường của $\mathbf{LAA}$ chạm điểm thấp nhất. Trong các hình này khoảng cách giữa các đường $\mathbf{RLA}$, DS, RS so với $\mathbf{LAA}$ trở nên lớn hơn khi $B \geq 1K$. Đối với Hept và Enron, đường $\mathbf{RLA}$ luôn nằm trên các đường còn lại, theo sau lần lượt là RS, DS và thấp nhất là $\mathbf{LAA}$. Khi $B \geq 1.5K$, đường DS đã tăng hơn so với RS và còn có thể tiến tới xấp xỉ $\mathbf{RLA}$. Điều này phản ánh đúng về lý thuyết khi $\mathbf{RLA}$ và DS có tỉ lệ xấp xỉ là tương đương nhau. Tuy nhiên, xu hướng chung giá trị hàm mục tiêu của $\mathbf{RLA}$ vẫn là tốt nhất. Điều này cho thấy chất lượng lời giải của $\mathbf{RLA}$ vượt trội hơn các thuật toán còn lại, điều này càng trở nên có ý nghĩa hơn khi n tăng, B tăng.

Thứ hai, Hình 2.4(b)-2.6(b) hiển thị số lượng truy vấn được gọi để chạy các



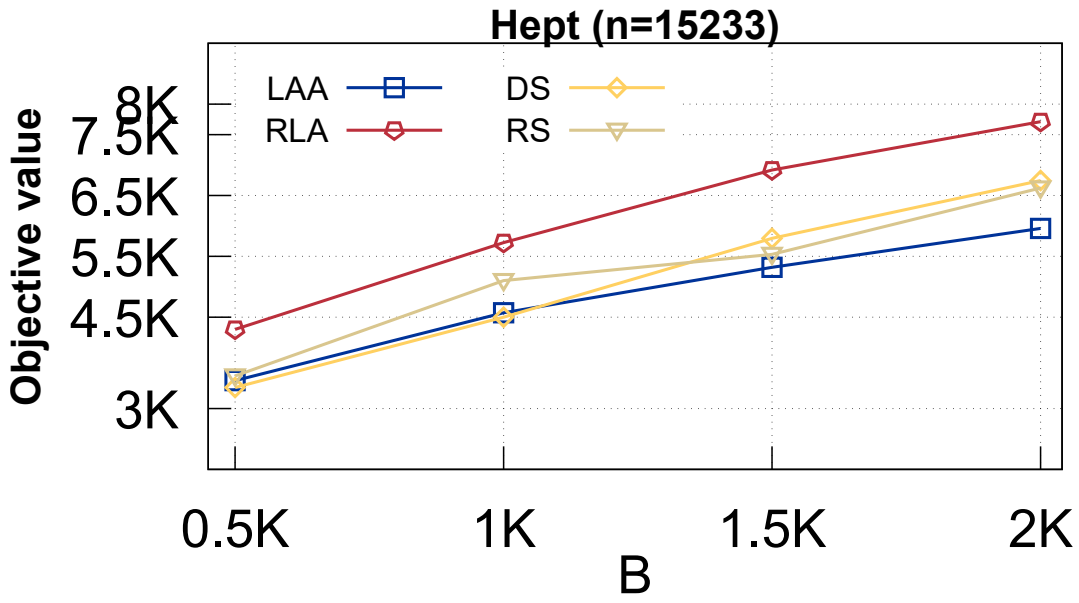
(a)



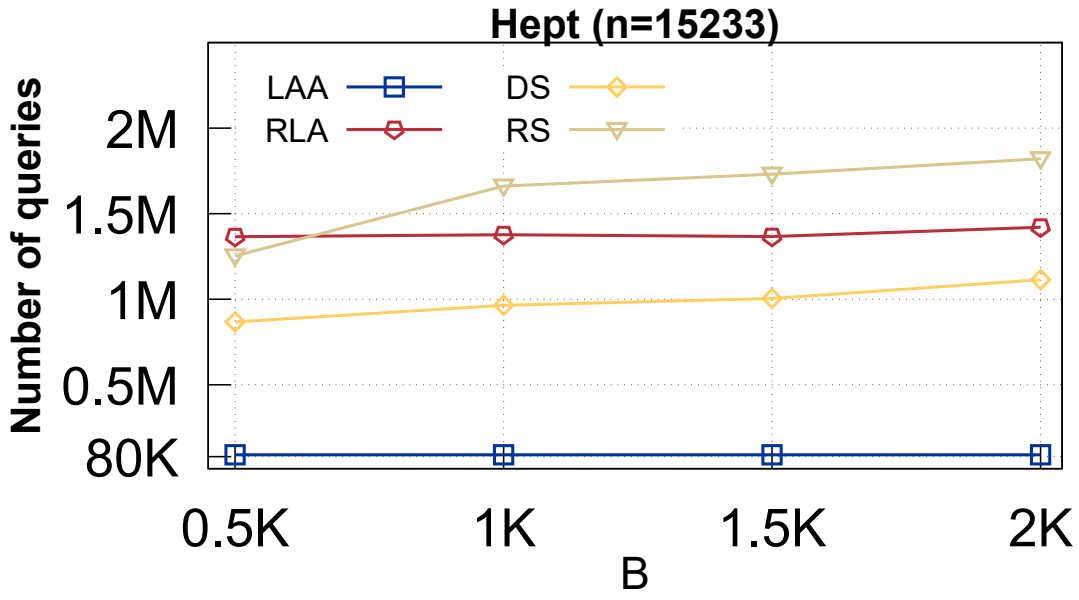
(b)

Hình 2.4: Các kết quả cho kIMK, trường hợp không đơn điệu trên dữ liệu Facebook: (a) Giá trị hàm mục tiêu, (b) Số lượng truy vấn.

thuật toán này. **LAA** tốn ít truy vấn nhất. Đối với Facebook và Hept, nó thấp hơn khoảng 8-25 lần, đối với Enron, nó thấp hơn khoảng 6-10 lần so với các thuật toán còn lại. Bên cạnh đó, đối với Facebook và Enron, số lượng truy vấn của **RLA** tương đương với DS và thấp hơn RS khoảng 3 lần đối với bộ dữ liệu nhỏ hơn là Facebook. Riêng với Hept, số truy vấn của nhóm RLA - DS - RS đang có sự phân hóa khi RS cao nhất, sau đó đến RLA va cuối cùng là DS. Tuy nhiên, có thể thấy



(a)

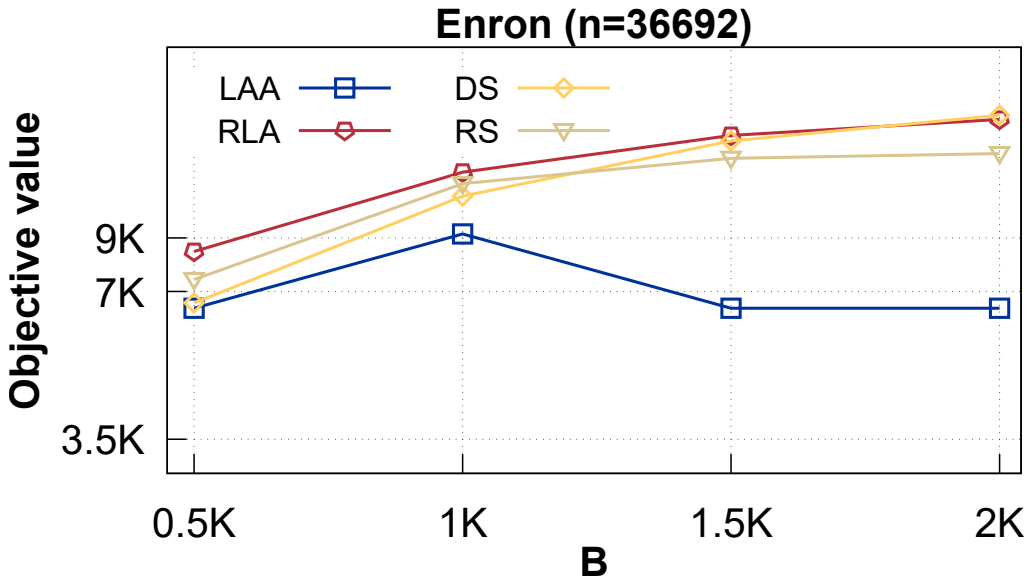


(b)

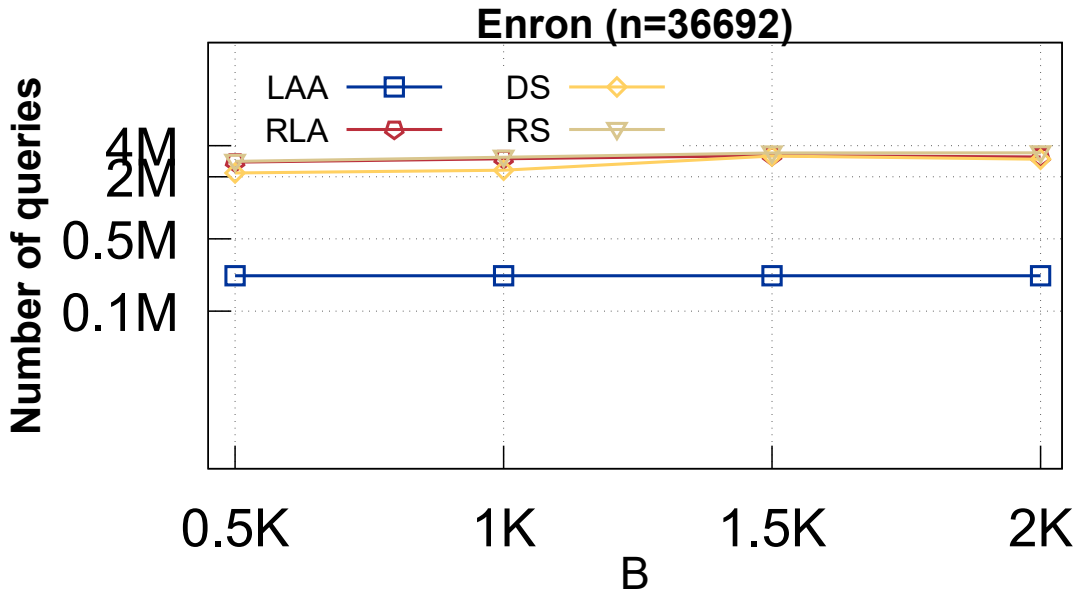
Hình 2.5: Các kết quả cho kIMK, trường hợp không đơn điệu trên dữ liệu Hept: (a) Giá trị hàm mục tiêu, (b) Số lượng truy vấn.

RLA đang có xu hướng tiến về gần DS. Một điều cần nhấn mạnh là xu hướng chung của các thuật toán này là nó giữ sự ổn định, đặc biệt là với **LAA**, **RLA** và DS. Vì đây là các thuật toán tất định. Thêm vào đó, khi n, B tăng lên, các thuật toán của luận án vẫn cho chất lượng lời giải vượt trội mà số lượng truy vấn nói chung là tốt tương đương với các thuật toán hiện nay.

Có thể thấy DS luôn giữ lượng truy vấn tốt so với các thuật toán còn lại đối



(a)



(b)

Hình 2.6: Các kết quả cho kIMK, trường hợp không đơn điệu trên dữ liệu Enron: (a) Giá trị hàm mục tiêu, (b) Số lượng truy vấn.

với mọi tập dữ liệu. Điểm mạnh của **RLA** là nó vẫn bám sát được DS, trong khi đó chất lượng lời giải lại luôn luôn tốt. Thậm chí khi n tăng (Hept, Enron), thuật toán này cho chất lượng tốt nhất. Sự đánh đổi thể hiện ở đây là **RLA** đã chấp nhận mất thêm một lượng truy vấn, nhưng vẫn đảm bảo độ phức tạp là gần tuyến tính (qua chứng minh lý thuyết và qua các đường ổn định trong thực nghiệm) nhưng chất lượng lời giải đảm bảo tốt nhất.

Về mặt lý thuyết, chất lượng lời giải của **RLA** và DS là tương đương nhưng số truy vấn được giảm hơn. Phần thực nghiệm được thực hiện không phải trong điều kiện lý tưởng khi các thuật toán do hệ thống phải chia sẻ tài nguyên. Tuy nhiên, thực nghiệm đã chỉ ra được sự tiệm cận giữa thực tiễn và lý thuyết; cho thấy sự cân bằng, đánh đối giữa tỉ lệ xấp xỉ và độ phức tạp truy vấn của các thuật toán. Ngoài ra, phần này cũng khẳng định thời gian chạy, độ phức tạp truy vấn của các thuật toán là tất định và tuyến tính. Điều này là rất quan trọng để các thuật toán nêu trong luận án có thể giải bài toán kSMK với dữ liệu có kích thước lớn.

2.8. Kết luận chương

Luận án đã trình bày các nghiên cứu với kSMK và đề xuất các thuật toán giải với trường hợp hàm mục tiêu đơn điệu và không đơn điệu. Đối với hàm mục tiêu đơn điệu, luận án đóng góp 03 thuật toán xấp xỉ, trong đó **IFA+** cho tỉ lệ xấp xỉ lên tới $1/3 - \epsilon$ với độ phức tạp tuyến tính, tốt hơn thuật toán xấp xỉ tốt nhất hiện nay. Đối với hàm mục tiêu không đơn điệu, tác giả đề xuất thuật toán xấp xỉ **RLA** cho tỉ lệ đạt $1/5 - \epsilon$, tương đương với thuật toán xấp xỉ tốt nhất hiện nay, nhưng giảm được số lượng truy vấn. Phần thực nghiệm so sánh giữa các thuật toán cho thấy sự tiệm cận về lý thuyết với thực tiễn, củng cố kết luận các thuật toán do luận án đóng góp cải tiến đáng kể các đảm bảo lý thuyết.

Các nghiên cứu góp phần thúc đẩy cải tiến thuật toán tham lam bằng phương pháp ngưỡng tham lam áp dụng cho các bài toán tối ưu hàm submodular khác. Các kết quả nghiên cứu của luận án đã được công nhận tại các bài báo:

1. *Fast Streaming Algorithms for k-Submodular Maximization under a Knapsack Constraint*, IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA), 2022 (**ISI/RANK A**);

2. *Improved Approximation Algorithms for k-Submodular Maximization under a Knapsack Constraint*, Computers & Operations Research, 2023 (**ISI/Q1**).

3. *Robust Approximation Algorithms for Non-monotone k-Submodular Maximization under a Knapsack Constraint*, the 15th IEEE International Conference on Knowledge and Systems Engineering (KSE 2023).

CHƯƠNG 3
BÀI TOÁN TỐI ĐA HÀM SUBMODULAR ĐƠN ĐIỆU VỚI RÀNG BUỘC CHI PHÍ CÓ NHIỄU

Ở chương trước, luận án đã tiếp cận hướng nghiên cứu mở rộng hàm mục tiêu từ submodular sang k-submodular với nhiều ý nghĩa lý thuyết và thực tiễn. Ta có thể thấy bài toán tối ưu tổ hợp hàm dạng submodular và k-submodular rất có giá trị khi ứng dụng trong học máy và trí tuệ nhân tạo khi ứng dụng được trong nhiều lĩnh vực như tối đa ảnh hưởng, đặt cảm biến, tối ưu tài nguyên... Tuy nhiên khi nghiên cứu về các bài toán này, rất ít tác giả đặt vấn đề ước lượng hàm mục tiêu bị chi phối bởi nhiễu. Trong khi đó, vấn đề với nhiễu là một vấn đề thường gặp và tiêu tốn rất nhiều chi phí để xử lý nó.

Đặt vấn đề nghiên cứu với nhiễu, luận án giải bài toán *tối đa hàm submodular với ràng buộc chi phí có nhiễu (Submodular Maximization subject to a Knapsack constraint under Noises - SMKN)*. Đây là một bài toán mới, hiện chưa có nghiên cứu nào giải quyết bài toán này. Thêm vào đó, độ khó của bài toán NP-khó và ràng buộc chi phí làm cho bài toán có tính thách thức hơn.

Ở chương này, luận án đã đề xuất hai thuật toán xấp xỉ là GUN, một phiên bản của thuật toán tham lam và NS là thuật toán luồng. Thuật toán luồng được thiết kế nhằm cải tiến thuật toán tham lam về vấn đề thời gian chạy và bộ nhớ. Bên cạnh đó, luận án cũng đưa ra các thực nghiệm đối với các thuật toán này.

3.1. Phát biểu bài toán, hàm mục tiêu và một số quy ước quan trọng

3.1.1. Phát biểu bài toán

Ràng buộc chi phí tạo ra một lớp các bài toán xuất phát từ thực tế cần giới hạn về số lượng, thời gian, ngân sách... Ví dụ khi thực hiện lan truyền tiếp thị, một công ty muốn lôi kéo người dùng đến mua hàng bằng cách tặng phiếu giảm giá cho những người dùng trên mạng xã hội. Họ chọn tặng cho một số người có tầm ảnh hưởng nhất định vì ngân sách là hữu hạn. Tương tự như vậy, khi muốn giám sát chất lượng nước hay chất lượng không khí trong một không gian hay bề mặt thì số lượng cảm biến cũng có giới hạn... Các bài toán như vậy đều có thể khái quát thành bài toán tối đa hàm submodular với ràng buộc chi phí. Qua đó cho thấy, các bài toán tối ưu có thêm ràng buộc chi phí sẽ đáp ứng được nhiều trường hợp thực tế. Bài toán *tối đa hàm submodular với ràng buộc chi phí (Submodular Maximization subject to Knapsack constraint - SMK)* có phát biểu như sau:

Định nghĩa 3.1 (Bài toán SMK)**.** Cho tập cơ sở V, mỗi phần tử $v \in V$ có một chi phí không âm, $c(v) \in \mathbb{R}_+$ cần tiêu tốn để phần tử này hoạt động theo mục

tiêu, gọi là chi phí để kích hoạt phần tử. Cho trước hàm $f : 2^V \mapsto \mathbb{R}_+$ là một hàm submodular không âm. Cho $B > 0$ là ngân sách cho trước. Bài toán yêu cầu cần tìm tập lời giải S sao cho $\max_{S \subseteq V : c(S) \leq B} f(S)$, với hàm chi phí là một hàm *modular*, tức là với mọi tập S, có $c(S) = \sum_{v \in S} c(v)$.

Bài toán SMK đang được quan tâm nhiều hiện nay [62, 70, 131, 88]... Tuy nhiên, các nghiên cứu thường bỏ qua một chi tiết là tính toán hàm f có thể rất phức tạp. Ví dụ với bài toán tối đa ảnh hưởng [43], IM, việc tính toán hàm ảnh hưởng đã được chứng minh là #P-khó [33, 29]. Thay vì tính toán chính xác hàm mục tiêu, nó được ước lượng bằng cách mô phỏng lại quá trình lan truyền thông tin ngẫu nhiên, tạo ra nhiễu (noise). Trong quá trình phân tích, chỉ một phần nhỏ của dữ liệu được sử dụng để đánh giá, tạo ra tính toán nhiễu (evaluation noisy).

Trong một nghiên cứu khác về tối đa hàm k-submodular với ràng buộc kích thước, Lan Nguyen [106] cũng chỉ ra hàm mục tiêu được định nghĩa với một tập cơ sở mà ta chưa rõ phân phối và ta chỉ có thể quan sát hoặc tìm được một hàm nhỏ các hàm tính được từ phân phối đó. Vì vậy, thay vì tính toán một cách chính xác hàm f, ta phải ước lượng F của f. F này được gọi là *ước lượng xấp xỉ (approximate oracle)* hay ước lượng nhiễu *ước lượng nhiễu (noise oracle)* ϵ của hàm mục tiêu.

Luận án đặt vấn đề xem xét bài toán SMK dưới sự tác động của nhiễu được mô hình hoá thành bài toán SMKN. Hai loại nhiễu được xem xét là *nhiễu cộng (additive noise)* và *nhiễu nhân (multiplicative noise)*. Bài toán này có định nghĩa như sau:

Định nghĩa 3.2 (Bài toán SMKN)**.** Bài toán SMK có nhiễu tồn tại một hàm F là ước lượng nhiễu của hàm mục tiêu f submodular đơn điệu. Gọi $\epsilon > 0$ là tham số nhiễu, hàm f dưới sự ảnh hưởng của nhiễu cộng hoặc nhiễu nhân được xấp xỉ theo ước lượng như sau:

- Với **nhiễu nhân** ϵ [69, 117, 147]) có:

$$(1 - \epsilon) f(S) \leq F(S) \leq (1 + \epsilon) f(S). \tag{3.1}$$

- Với **nhiễu cộng** ϵ [40, 147]), có:

$$f(S) - \epsilon \leq F(S) \leq f(S) + \epsilon. \tag{3.2}$$

3.1.2. Hàm mục tiêu và một số quy ước quan trọng

Để thuận tiện trong việc đưa ra các phát biểu tính chất hàm mục tiêu, xây dựng các thuật toán và phân tích lý thuyết, Bảng 3.1 tóm tắt các ký hiệu thường

dùng khi giải bài toán này.

Ký hiệu	Mô tả
V	Tập cơ sở $V = \{e_1, e_2, .., e_n\}$
n	Kích cỡ của tập V
$c(e)$	Chi phí của một đỉnh $e \in V$ bất kỳ
S	Tập các phần tử
B	Ngân sách cho trước, $B \in \mathbb{Z}_+$
$\Delta(e\|A)$	Lợi nhuận biên (marginal gain) khi thêm một phần tử e vào tập S
$f(A+e)$	Cách viết tắt của $f(A \cup \{e\})$
S^*, opt	S^* là lời giải tối ưu, opt $= f(S^*)$
γ	$\gamma \in (0, 1)$ là một tham số chính xác cho trước của thuật toán
ϵ	$\epsilon \in (0, 1)$ là tham số nhiễu cho trước của thuật toán
m	$m = \max_{e \in V} f(e)$ là giá trị lớn nhất của một phần tử (maximal singleton value)

Bảng 3.1: Các ký hiệu thường dùng trong bài toán SMKN

Cụ thể, ta sử dụng ký hiệu $\Delta(e|A)$ là lợi nhuận biên. Ngoài ra, để đơn giản, các phép toán tính hàm $f(\cdot)$ khi thêm 1 phần tử e vào trong một tập S nào đó, công thức $f(S \cup \{e\})$ được viết thành $f(S + e)$. Như vậy lợi nhuận biên khi thêm một phần tử e vào một tập A là: $\Delta(e|A) = f(A + e) - f(A)$. Trong chương này, để giải quyết bài toán, ta cần sử dụng các tính chất của hàm mục tiêu như sau:

- Tính chất lợi nhuận hiệu suất giảm dần tự nhiên của hàm submodular

Tính chất lợi nhuận hiệu suất giảm dần tự nhiên của hàm mục tiêu phù hợp với các phân tích cho hàm submodular rời rạc. Lý do là vì tính chất hiệu suất giảm dần chỉ rõ được đóng góp lợi ích của một phần tử đối với một tập. Ngoài ra, nó còn kết hợp được với tính chất đơn điệu để tìm ra các giới hạn khi phân tích giá trị của một tập. Do vậy, nó phù hợp để thiết kế và phân tích các thuật toán.

Nhắc lại, tính chất này phát biểu rằng với các tập $A \subseteq B \subseteq V$, với mọi $e \notin B$ ta có lợi nhuận biên khi thêm một phần tử vào một tập hợp thỏa mãn:

$$\Delta(e|A) = f(A + e) - f(A) \geq f(B + e) - f(B) = \Delta(e|B). \tag{3.3}$$

Như vậy, tính chất này thể hiện rằng lợi ích thu được khi thêm vào tập nhỏ hơn ít nhất bằng lợi ích thu được khi thêm vào tập lớn hơn.

- Hàm đơn điệu tăng

Luận án giới hạn phạm vi giải bài toán SMKN với giả thiết hàm f là hàm đơn điệu tăng. Tính chất đơn điệu tăng thoả mãn bất đẳng thức sau:

$$f(A) \leq f(B) \;\; \forall A \subseteq B. \subseteq V \tag{3.4}$$

Bên cạnh đó, trong tính toán, các thuật toán vẫn sử dụng giả thiết tồn tại một

hộp đen để với mọi tập A, đều trả về giá trị của $f(A)$. Nó gắn liền với bài toán luôn giả sử đã cho hàm $f(\cdot)$. Thêm vào đó, hàm f được chuẩn hóa, $f(\emptyset) = 0$, tức là giá trị của tập rỗng là không có ý nghĩa đóng góp vào lời giải. Đồng thời, ta ký hiệu S^* là lời giải tối ưu, giá trị tối ưu là $\mathsf{opt} = f(S^*)$.

3.2. Các thách thức của bài toán

Với bài toán SMKN, đây là một bài toán mới, việc đề xuất các thuật toán xấp xỉ hiệu quả nhằm giải quyết bài toán cần phải đối mặt với nhiều thách thức, có thể kể đến như sau:

- Thách thức của ràng buộc chi phí. Với ràng buộc này các phần tử có chi phí khác nhau, làm cho chi phí lời giải khác nhau với điều kiện không vượt quá B. Do vậy, bài toán sẽ có nhiều lời giải dự tuyển khác nhau. Vì thế thời gian tính toán là khó xác định.

- Thách thức về khả năng mở rộng của thuật toán. Với vấn đề dữ liệu lớn, không gian tìm kiếm lời giải trở nên khổng lồ, rất cần các thuật toán xấp xỉ hiệu quả vừa giảm dung lượng bộ nhớ lưu trữ, thời gian chạy vừa đảm bảo chất lượng lời giải.

- Thách thức khi xử lý bài toán SMK với nhiễu. Khi giải bài toán SMKN sẽ gặp nhiều khó khăn hơn vì hàm F có thể không thừa kế lại các tính chất của hàm f để có thể vận dụng lại các kết quả khi phân tích một thuật toán cho SMK. Thực tế, một vài tác giả đã chỉ ra rằng F có thể không còn là đơn điệu hoặc không còn là submodular trong nhiều tình huống [106, 147]. Thêm vào đó, sự khác nhau về mặt bản chất giữa hàm f và F dẫn tới các hiệu quả không rõ ràng khi áp dụng trực tiếp các thuật toán đã có với SMK cho SMKN. Cuối cùng, việc phân tích các đảm bảo hay các giới hạn cho lời giải khi chỉ sử dụng ước lượng nhiễu cũng rất khó khăn.

- Thách thức về xây dựng ước lượng F của hàm mục tiêu phù hợp với từng bài toán ứng dụng cụ thể. Hiện nay, mới có cách tính ước lượng F dựa trên *lưới ảnh hưởng ngược trung bình (average reachability sketch)* đề xuất bởi Cohen và cộng sự [36] cho bài toán lan truyền thông tin giải quyết ước lượng F có thể không còn là submodular.

Tóm lại, chưa có tác giả nào đặt vấn đề nghiên cứu bài toán SMK với nhiễu và đề xuất thuật toán khả thi để giải. Do vậy, luận án giải bài toán *tối đa hàm submodular với ràng buộc chi phí có nhiễu,* SMKN, bằng các thuật toán xấp xỉ hiệu quả. Các thách thức trên đây vừa là khó khăn vừa là động lực để luận án tiến hành nghiên cứu bài toán SMKN theo các tiêu chí: giảm thời gian chạy của thuật toán, giảm dung lượng lưu trữ cần cho các phần tử và cải thiện tỉ lệ xấp xỉ có thể cạnh tranh được với các thuật toán tham lam.

Để có thể đề xuất các thuật toán hiệu quả giải bài toán này, việc tìm hiểu các

kỹ thuật hiện nay là cấp thiết. Phần tiếp theo đề cập đến các nghiên cứu có liên quan, bao gồm các nghiên cứu đối với bài toán SMK và các nghiên cứu áp dụng mô hình nhiễu vào trong bài toán tối ưu hàm submodular.

3.3. Các vấn đề nghiên cứu có liên quan

Bài toán SMK cần tìm tập lời giải S với chi phí không vượt quá ngân sách B tức là $c(S) \leq B$. Như vậy, bài toán có thể cho rất nhiều lời giải cho chi phí khác nhau và kích thước lời giải cũng khác nhau. Nemhauser và Wolsey [146] khởi xướng nghiên cứu bài toán SMK và chỉ ra rằng bài toán này là NP-khó. Các tác giả đề xuất một thuật toán tham lam xấp xỉ tuần tự chọn phần tử e nào chưa được chọn từ tập cơ sở V có tỉ lệ $\Delta(e|S)/c(e)$ đạt giá trị lớn nhất. Các tác giả cũng chỉ ra rằng khi lời giải của thuật toán tham lam đạt được chính xác ngân sách B, thuật toán đảm bảo lời giải xấp xỉ $(1-1/e) \approx 63\%$ lời giải tối ưu. Tuy nhiên, công trình của Nemhauser và Wolsey không chỉ ra độ phức tạp tính toán là bao nhiêu.

Sau đó, các tác giả khác [131, 92] đề xuất các lời giải gần đạt tỉ lệ $(1 - 1/e)$ nhưng các thuật toán mà họ đề xuất có độ phức tạp truy vấn lên tới $O(n^5)$. Leskovec và cộng sự [92] giới thiệu một cách tiếp cận mới đạt được tỉ lệ xấp xỉ $(1 - 1/e)$ cho trường hợp chi phí đồng nhất, tức là mọi phần tử đều có chi phí là như nhau, và tỉ lệ $(e - 1)/2e$ trong trường hợp chi phí không đồng nhất, nhưng thuật toán chạy nhanh hơn 700 lần so với tham lam đơn giản. Các tác giả khác như [88, 28, 21, 48] đã cố gắng cải tiến tỉ lệ xấp xỉ nói trên thành $(1 - 1/e + \epsilon)$ hoặc $(1/e + \epsilon)$, với $\epsilon > 0$ gọi là hằng số chính xác, và cũng mở rộng bài toán cho trường hợp tổng quát là hàm f có thể đơn điệu hoặc không. Tuy nhiên, các nghiên cứu của họ phải xử lý trường hợp mở rộng hàm f thành hàm tuyến tính liên tục phức tạp và vẫn cần lần lượt $O(n^c)$, $O(n \log^2 n)$, $O(n)$ và $O(n \log n)$ lời gọi hàm f. Khi sử dụng phương pháp này, cách tính hàm f thường là phức tạp, cần nhiều truy vấn có thể dẫn tới kết quả xấu tùy ý khó lường trước [8].

Nhằm cải tiến các thuật toán, giảm thời gian chạy, một số tác giả đã sử dụng các phiên bản thuật toán luồng khác nhau [70, 67], các tác giả khác nghiên cứu xây dựng các thuật toán song song [27, 11]. Ngoài ra, có một số tác giả xem xét bài toán với hàm mục tiêu có thể không đơn điệu [7, 51, 49, 86]...

Với bài toán SMK đơn điệu, Sviridenko [131] đã đề xuất thuật toán tham lam để giải. Sau đó Gupta và cộng sự [65] đã đề xuất thuật toán cải tiến dựa trên công bố của Sviridenko, cho tỉ lệ xấp xỉ là $\alpha + 1/2(1 - 1/e)$ cho trường hợp chi phí các đỉnh là 1, và tỉ lệ xấp xỉ là $(1/4 + \alpha)$ khi f là không đơn điệu. Tuy nhiên độ phức tạp truy vấn trong thuật toán của họ là $O(n^4)$. Số lượng truy vấn này rất lớn, nên khi tập V lớn, số lần gọi hàm f tăng làm ảnh hưởng tới tốc độ thuật toán rất nhiều.

Gần đây, các tác giả như Amanatidis và cộng sự [1, 2], Han và cộng sự [67] đã xây dựng các thuật toán luồng, thiết kế song song, ngưỡng tham lam giảm dần để đạt được tỉ lệ xấp xỉ là tất định tốt nhất với lượng truy vấn và số vòng lặp tuần tự chỉ còn là tuyến tính hoặc gần tuyến tính. Các công bố gần đây đưa rất nhiều các phương pháp thiết kế khác nhau nhằm cải tiến giảm số lượng truy vấn, giảm thời gian chạy bên cạnh việc cải thiện tỉ lệ xấp xỉ. Đặc biệt, xu hướng xây dựng các thuật toán xấp xỉ cho tỉ lệ là hằng số đang được chú trọng.

Cụ thể, Amanatidis và cộng sự [1] đề xuất thuật toán song song lần đầu tiên đạt được tỉ lệ xấp xỉ tất định là $(1/5.83 - \epsilon)$ với trường hợp hàm mục tiêu không đơn điệu, cho độ phức tạp song song gần tối ưu là $O(\log n)$ và độ phức tạp truy vấn là $O(\frac{n}{\epsilon}\log\frac{n}{\epsilon})$. Đặc biệt, Han và cộng sự [67] đã thiết kế các thuật toán ngẫu nhiên và luồng để giải bài toán SMK. Thuật toán ngẫu nhiên của họ cho tỉ lệ xấp xỉ $1/4$ và độ phức tạp truy vấn là $O(kn)$, thuật toán ngẫu nhiên cải tiến cho tỉ lệ là $(1/4-\epsilon)$ và độ phức tạp truy vấn là $O(\frac{n}{\epsilon}\log(\frac{k}{\epsilon}))$. Trong khi đó, thuật toán luồng lại lần lượt cho tỉ lệ $(1/6 - \epsilon)$ and $(1/8 - \epsilon)$ với độ phức tạp truy vấn là $O(\frac{k}{\epsilon}\log(B))$. Cùng với đó, sử dụng cách tiếp cận luồng, các tác giả trong [70] cũng đề xuất 2 thuật toán trả về tỉ lệ lần lượt là $(0.363 - \epsilon)$ và $(0.4 - \epsilon)$.

Từ bối cảnh nêu trên, có thể thấy nghiên cứu về bài toán SMK là đang là chủ đề được quan tâm rộng rãi. Đồng thời, thiết kế thuật toán luồng giải bài toán này đang là xu hướng được quan tâm hiện nay.

Bên cạnh đó, việc xem xét giải bài toán tối đa hàm submodular trong môi trường nhiễu đã được đặt ra [124, 61, 96]. Môi trường có nhiễu là thường gặp trong tự nhiên. Ví dụ khi tiến hành lấy mẫu đo từ các cảm biến, ảnh chụp các đối tượng ngoài tự nhiên..., nhiễu sẽ ảnh hưởng rất nhiều đến kết quả, độ chính xác và thời gian tính toán [120] của thuật toán, vì thế rất cần các thuật toán hiệu quả để giải quyết bài toán vẫn đảm bảo chất lượng lời giải.

Đầu tiên, Qian và cộng sự đã xem xét bài toán trích chọn đặc trưng (một thể hiện của bài toán SMC) khi có có sự tác động của nhiễu cộng và nhiễu nhân [117]. Các tác giả đề xuất phương pháp giải dựa trên thuật toán tham lam. Sau đó, các tác giả trong [69] đã xây dựng mô hình tổng quát cho bài toán tối ưu hàm submodular với nhiễu. Một số tác giả [147] đã xây dựng thuật toán luồng nhằm giải quyết bài toán tối đa hàm submodular với nhiễu. Crawford và cộng sự [40] đã giải bài toán Tập phủ trong môi trường có nhiễu nhân và đề xuất một thuật toán tham lam cải tiến bằng cách tính toán dựa trên ước lượng nhiễu F thay vì hàm f. Sau này, Lan Nguyen và cộng sự [106] đã xây dựng các thuật toán luồng nhằm giải quyết bài toán tối đa hàm k-submodular với ràng buộc lực lượng. Tuy nhiên, nghiên cứu bài toán SMK trong môi trường có nhiễu lại chưa được đề cập.

Từ bối cảnh nghiên cứu trên cho thấy giải bài toán tối đa hàm submodular với ràng buộc chi phí trong môi trường có nhiễu là một bài toán quan trọng phù hợp với xu thế nghiên cứu hiện nay. Đây là động lực để NCS nghiên cứu giải bài toán này và đề xuất một thuật toán luồng 1 lần quét với độ phức tạp truy vấn $O(n \log B)$.

3.4. Thuật toán xấp xỉ cho bài toán SMKN

3.4.1. Kết quả mới của luận án

Đầu tiên, luận án đề xuất một phiên bản thuật toán của tham lam xấp xỉ với điều kiện đã biết ước lượng xấp xỉ F của hàm mục tiêu. Thuật toán tiếp theo là phiên bản cải tiến được thiết kế trở thành thuật toán luồng khi biết ước lượng xấp xỉ F của hàm mục tiêu. Mục đích của phiên bản đầu tiên là thiết kế một thuật toán tham lam làm cơ sở ban đầu cho bài toán SMKN. Phiên bản thứ hai cải tiến thuật toán tham lam, giảm thời gian tính toán và dung lượng lưu trữ với chất lượng lời giải vẫn được đảm bảo. Vì SMKN là một bài toán mới, nên phần thực nghiệm sẽ trực tiếp so sánh và đánh giá hai thuật toán này.

Nhìn chung, nghiên cứu về bài toán SMKN, luận án tập trung vào:

- Đề xuất thuật toán tham lam xấp xỉ giải quyết bài toán SMKN;
- Đề xuất thuật toán luồng cải tiến thuật toán tham lam nhằm duy trì tỉ lệ xấp xỉ tương đương nhưng vẫn giảm thời gian chạy và bộ nhớ lưu trữ các phần tử;
- Tiến hành thực nghiệm so sánh hai thuật toán đã đề xuất.

Luận án giải quyết trường hợp hàm mục tiêu là submodular đơn điệu. Thứ hai, luận án đưa các mô hình nhiễu cộng và nhiễu nhân và các ước lượng nhiễu vào trong thuật toán. Các phần tử được lựa chọn được đánh giá dựa trên tỉ lệ giữa lợi nhuận biên và chi phí của các phần tử đó. Thuật toán luồng cải tiến tạo nhiều lời giải dự tuyển và thiết kế một ngưỡng "sàng" nhằm chỉ giữ lại các phần tử tốt.

Tóm lại, những đóng góp của luận án này đối với bài toán SMKN đơn điệu tăng có thể kể đến là:

- Đề xuất thuật toán tham lam, **GUN** (Greedy Under Noises, Thuật toán 7), cho lời giải $f(S) \geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\left(1-\frac{1}{e}\right)\left(1-\frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt}$ đối với nhiễu nhân, và $f(S) \geq \frac{1}{2}\left(1-\frac{1}{e}\right)\mathsf{opt} - 2\epsilon(B+1)$ đối với nhiễu cộng. Thuật toán đưa vào ước lượng nhiễu ϵ, F, của f và chọn ra các phần tử đảm bảo tỉ lệ giữa đóng góp lợi ích và chi phí là lớn nhất.
- Đề xuất thuật toán luồng xấp xỉ, **NS** (Noise-processed Streaming, Thuật toán 9), cho chất lượng lời giải đạt $f(S) \geq (\frac{1-\epsilon}{1+\epsilon})^2 \frac{(1-\gamma)}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}\mathsf{opt}$ với nhiễu nhân và $f(S) \geq \frac{1-\gamma}{3}\mathsf{opt} + \epsilon - 2\epsilon B$ với nhiễu cộng, trong đó $\gamma \in (0,1)$ là một số cho trước. Đặc biệt, thuật toán này đã chỉ ra được độ phức tạp truy vấn và độ phức

tạp bộ nhớ đối với nhiễu nhân lần lượt là $O(\frac{n}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$ và $O(\frac{B}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$. Đối với nhiễu cộng lần lượt cần yêu cầu độ phức tạp truy vấn là $O(\frac{n}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon}))$ và độ phức tạp bộ nhớ là $O\left(n'\log(nB)\right)$ $O(\frac{B}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon})))$ $O(kn/\epsilon)$ với $\epsilon \in (0,1)$ là tham số nhiễu cho trước, $m = max_{e\in V} f(e)$. Thuật toán được xây dựng dựa trên một tập O giới hạn số phần tử v là ước lượng của opt và xây dựng S_v các lời giải dự tuyển tương ứng. Mỗi S_v sẽ có một ngưỡng sàng chọn ra các phần tử tốt.

- Tiến hành thực nghiệm trên một trường hợp cụ thể của SMKN với ứng dụng tối đa ảnh hưởng với ràng buộc chi phí (Influence Maximization under Knapsack constraint - IMK) trên 2 bộ dữ liệu hàng nghìn đỉnh và vài chục nghìn cạnh. Kết quả thực nghiệm cho thấy sự hội tụ giữa 2 thuật toán về giá trị hàm mục tiêu, nhưng **NS** tiết kiệm thời gian hơn nhiều so với **GUN**. Đặc biệt, số truy vấn của **NS** có trường hợp **thấp hơn tới 266 lần** so với **GUN**. Thêm vào đó, luận án cũng tiến hành đo bộ nhớ tối đa tập O cần sử dụng. So sánh cho thấy thay vì phải cấp phát bộ nhớ cho các phần tử trong tập V trong suốt quá trình tính toán, thuật toán yêu cầu một lượng giới hạn các phần tử ứng với tập O nhỏ hơn rất nhiều so với tập V, đặc biệt với B và V lớn.

Bảng 3.2 so sánh **GUN** và **NS** theo 3 khía cạnh: chất lượng lời giải, độ phức tạp truy vấn, và độ phức tạp bộ nhớ. $m = \max_{e\in V} F(e)$, $\gamma \in (0,1)$ là tham số đầu vào, $\epsilon > 0$ là tham số nhiễu. TT là viết tắt của thuật toán. Hai thuật toán **GUN** và **NS** cho chất lượng lời giải trong môi trường nhiễu là không chênh lệch quá lớn. Với **GUN**, các độ phức tạp truy vấn và bộ nhớ là $O(n^2)$, trong khi với **NS**, độ phức tạp bộ nhớ ít hơn so với độ phức tạp truy vấn một hệ số là n. Bộ nhớ trong **NS** phụ thuộc chủ yếu vào ngân sách B.

TT	Đảm bảo lời giải	Độ phức tạp truy vấn	Độ phức tạp bộ nhớ
Cho nhiễu nhân ϵ			
GUN	$f(S) \geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\left(1-\frac{1}{e}\right)\left(1-\frac{4\epsilon B}{1-\epsilon^2}\right)$ opt	$O(n^2)$	$O(n^2)$
NS	$f(S) \geq (\frac{1-\epsilon}{1+\epsilon})^2 \frac{(1-\gamma)}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$opt	$O(\frac{n}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$	$O(\frac{B}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$
Cho nhiễu cộng ϵ			
GUN	$f(S) \geq \frac{1}{2}\left(1-\frac{1}{e}\right)$ opt $-2\epsilon(B+1)$	$O(n^2)$	$O(n^2)$
NS	$f(S) \geq \frac{1-\gamma}{3}$opt $+\epsilon - 2\epsilon B$	$O(\frac{n}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon}))$	$O(\frac{B}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon})))$

Bảng 3.2: So sánh các thuật toán xấp xỉ **GUN** và **NS** cho SMKN.

3.4.2. Thuật toán tham lam xấp xỉ: GUN

Với bài toán SMKN, đầu tiên, ta thiết kế một thuật toán tham lam dành cho nó. Luận án giới thiệu phiên bản thuật toán tham lam, **GUN** (*Greedy Under Noises*),

được thiết kế dựa trên ý tưởng chọn các phần tử cho tỉ lệ đóng góp lợi nhuận biên và chi phí của phần tử đó là lớn nhất trong các phần tử chưa được chọn. Ý tưởng này ban đầu được đề xuất bởi các tác giả trong [76] khi giải bài toán Tập phủ tối đa. Tuy nhiên, việc áp dụng giải cho SMKN cần một vài biến đổi. Thứ nhất, GUN sử dụng ước lượng F để thiết lập mối quan hệ giữa lời giải tối ưu với lời giải dự tuyển đang xét. Thứ hai, mô hình nhiễu cộng và nhiễu nhân được đưa vào để đáp ứng sự tác động của nhiễu đối với hàm mục tiêu. Chi tiết của thuật toán được giới thiệu trong Thuật toán 7.

Algorithm 7 : GUN

Input: V, an approximation oracle F, $\epsilon \in (0, 1)$, budget $B > 0$
Output: A subset S

$S \leftarrow \emptyset, U \leftarrow V$
if F: ϵ-multiplicative noise oracle **then**
  for $U = \emptyset$ **do**
    $e' \leftarrow \arg\max_{e \in U} \frac{1}{c(e)} \left(\frac{F(S+e)}{1-\epsilon} - \frac{F(S)}{1+\epsilon} \right)$
    if $c(S + e') \leq B$ **then**
      $S \leftarrow S + e'$
    end if
    $U \leftarrow U \setminus \{e'\}$
  end for
end if
if F: ϵ-additive noise oracle **then**
  for $U = \emptyset$ **do**
    $e' \leftarrow \arg\max_{e \in U} \frac{1}{c(e)} \left(F(S + e) - F(S) \right)$
    if $c(S + e') \leq B$ **then**
      $S \leftarrow S + e'$
    end if
    $U \leftarrow U \setminus \{e'\}$
  end for
end if
$e_m \leftarrow \arg\max_{e \in V} F(e)$
$S \leftarrow \arg\max_{S' \in \{S, e_m\}} F(S')$
return S

Thuật toán khởi tạo một lời giải dự tuyển S là một tập rỗng và tập các phần tử ứng viên $U = V$. Với pha đầu tiên, thuật toán lần lượt lựa chọn một phần tử e' thỏa mãn:

$$e' = \arg\max_{e \in U} \frac{1}{c(e)} \left(\frac{F(S + e)}{1 - \epsilon} - \frac{F(S)}{1 + \epsilon} \right)$$

với nhiễu nhân ϵ, và pha 2 làm việc với nhiễu cộng ϵ:

$$e' = \arg\max_{e \in U} \frac{1}{c(e)} \left(F(S+e) - F(S) \right).$$

Sau đó, e' được thêm vào lời giải hiện tại với điều kiện thêm e' vào tập S vẫn không vượt quá ngân sách $c(S+e') \leq B$. Công việc trên sẽ tiến hành lặp đi lặp lại cho tới khi tập phần tử ứng viên rỗng. Chúng ta có thể thấy thuật toán đã đưa vào một hệ số gọi là tỉ lệ lớn nhất giữa lợi nhuận biên so với chi phí của một phần tử e khi thêm vào tập S:

Với nhiễu nhân:

$$\frac{1}{c(e)} \left(\frac{F(S+e)}{1-\epsilon} - \frac{F(S)}{1+\epsilon} \right).$$

Với nhiễu cộng:

$$\frac{1}{c(e)} \left(F(S+e) - F(S) \right).$$

Mục tiêu của thuật toán là tìm phần tử e nào làm cho tỉ lệ trên là lớn nhất. Đồng thời, thuật toán cũng chọn ra một phần tử $e_m = \max_{e \in V} F(e)$ và trả lại lời giải tốt nhất là lời giải cho giá trị $F(\cdot)$ lớn nhất giữa S và e_m. Thêm vào đó, không mất tính tổng quát, có thể giả sử mọi phần tử $e \in V$ đều có chi phí $c(e) \leq B$, vì nếu có phần tử nào thực sự vượt hơn, đơn giản ta có thể bỏ qua phần tử đó.

Để thiết lập mối quan hệ giữa lời giải của thuật toán **GUN** và lời giải tối ưu, ta có Bổ đề 3.1.

Bổ đề 3.1. *Đặt $S_i = \{s_1, s_2, \ldots, s_i\}$ là lời giải ứng viên của thuật toán* **GUN** *tại bước lặp thứ i của vòng lặp chính.Với nhiễu nhân ϵ, chúng ta có:*

$$\mathsf{opt} - f(S_i) \leq \frac{B}{c(s_{i+1})} \left(f(S_{i+1}) - f(S_i) \right) + \frac{4\epsilon B}{1-\epsilon^2} \mathsf{opt}. \tag{3.5}$$

Và với nhiễu cộng ϵ:

$$\mathsf{opt} - f(S_i) \leq \frac{B}{c(s_{i+1})} \left(f(S_{i+1}) - f(S_i) \right) + 4\epsilon B. \tag{3.6}$$

Chứng minh. Đặt $S_i' = S^* \setminus S_i$. Với nhiễu nhân, ta có:

$$
\begin{aligned}
\mathsf{opt} - f(S_i) &\leq f(S_i' \cup S_i) - f(S_i) \\
&\leq \sum_{e \in S_i'} (f(S_i + e) - f(S_i)) \text{ (Vì tính submodular của } f) \\
&\leq \sum_{e \in S_i'} \left(\frac{F(S_i + e)}{1 - \epsilon} - \frac{F(S_i)}{1 + \epsilon} \right) (F \text{ là ước lượng với nhiễu nhân } \epsilon \text{ của } f) \\
&\leq \frac{B}{c(S_i')} \sum_{e \in S_i'} \left(\frac{F(S_i + e)}{1 - \epsilon} - \frac{F(S_i)}{1 + \epsilon} \right) \\
&\leq B \max_{e \in S_i'} \frac{1}{c(e)} \left(\frac{F(S_i + e)}{1 - \epsilon} - \frac{F(S_i)}{1 + \epsilon} \right) \text{ (Do sự lựa chọn của thuật toán)} \\
&= \frac{B}{c(s_{i+1})} \left(\frac{F(S_{i+1})}{1 - \epsilon} - \frac{F(S_i)}{1 + \epsilon} \right) \\
&\leq \frac{B}{c(s_{i+1})} \left(\frac{1 + \epsilon}{1 - \epsilon} f(S_{i+1}) - \frac{1 - \epsilon}{1 + \epsilon} f(S_i) \right) \\
&\leq \frac{B}{c(s_{i+1})} \left(f(S_{i+1}) - f(S_i) + \frac{2\epsilon}{1 - \epsilon} f(S_{i+1}) + \frac{2\epsilon}{1 + \epsilon} f(S_i) \right) \\
&\leq \frac{B}{c(s_{i+1})} (f(S_{i+1}) - f(S_i)) + \frac{4\epsilon B}{1 - \epsilon^2} \mathsf{opt}
\end{aligned}
$$

(Vì $c(s_{i+1}) \geq 1$) và $f(S_i) \leq \mathsf{opt}$).

Tương tự, với nhiễu cộng ta có:

$$
\begin{aligned}
\mathsf{opt} - f(S_i) &\leq \sum_{e \in S_i'} (f(S_i + e) - f(S_i)) \\
&\leq \sum_{e \in S_i'} (F(S_i + e) - F(S_i) + 2\epsilon) \\
&\leq \frac{B}{c(s_{i+1})} (F(S_{i+1}) - F(S_i) + 2\epsilon) \\
&\leq \frac{B}{c(s_{i+1})} (f(S_{i+1}) - f(S_i) + 4\epsilon) \\
&\leq \frac{B}{c(s_{i+1})} (f(S_{i+1}) - f(S_i)) + 4\epsilon B.
\end{aligned}
$$

Ta có bổ đề được chứng minh. □

Chất lượng lời giải của thuật toán trên được thể hiện qua Định lý 3.1:

Định lý 3.1. *Thuật toán* GUN *cho lời giải* S *đảm bảo:*

$$f(S) \geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\left(1-\frac{1}{e}\right)\left(1-\frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt}$$

cho nhiễu nhân ϵ *và*

$$f(S) \geq \frac{1}{2}\left(1-\frac{1}{e}\right)\mathsf{opt} - 2\epsilon(B+1)$$

cho nhiễu cộng ϵ*.*

Chứng minh. Đặt t là bước lặp cuối cùng của vòng lặp chính của thuật toán, và s_{t+1} là phần tử đầu tiên được thêm vào S_t nếu chúng ta bỏ qua điều kiện và tổng chi phí (dòng 4 hoặc 14) của Thuật toán 7.

Với nhiễu nhân, ta xem xét 2 trường hợp sau đây:

- Nếu $f(S_{t+1}) \geq \mathsf{opt}$, ta có:

$$\begin{aligned} f(S) &\geq \frac{1}{1+\epsilon}\max\{F(S_t), F(e_m)\} \text{ (Theo dòng 21 của thuật toán)} \\ &\geq \frac{1}{1+\epsilon}\max\{F(S_t), F(s_{t+1})\} \geq \frac{1-\epsilon}{1+\epsilon}\max\{f(S_t), f(s_{t+1})\} \\ &\geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}f(S_{t+1}) \text{ (Do tính submodular của } f) \\ &\geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\mathsf{opt}. \end{aligned}$$

- Nếu $f(S_{t+1}) < \mathsf{opt}$, tương tự với cách phân tích của Bổ đề 3.1, với $i \leq t$ bất kì ta có:

$$\mathsf{opt} - f(S_i) \leq \frac{B}{c(s_{i+1})}\left(f(S_{i+1}) - f(S_i)\right) + \frac{4\epsilon B}{1-\epsilon^2}\mathsf{opt}.$$

Bằng cách sắp xếp các bất đẳng thức trên và áp dụng một phương pháp suy diễn với t, ta có:

$$\begin{aligned} f(S_{t+1}) - (1-\frac{4\epsilon B}{1-\epsilon^2})\mathsf{opt} &\geq \left(1-\frac{c(s_{t+1})}{B}\right)\left(f(S_t) - (1-\frac{4\epsilon B}{1-\epsilon^2})\mathsf{opt}\right) \\ &\geq \ldots \geq -\prod_{i=0}^{t}\left(1-\frac{c(s_{i+1})}{B}\right)\left(1-\frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt}. \end{aligned}$$

Có nghĩa là:

$$\begin{aligned} f(S_{t+1}) &\geq \left(1 - \prod_{i=1}^{t+1}(1 - \frac{c(s_i)}{B})\right)\left(1 - \frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt} \\ &\geq \left(1 - \prod_{i=1}^{t+1}(1 - \frac{c(s_i)}{c(S_{t+1})})\right)\left(1 - \frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt} \\ &\geq \left(1 - \frac{1}{e}\right)\left(1 - \frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt}. \end{aligned}$$

Vì vậy: $f(S) \geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}f(S_{t+1}) \geq \frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\left(1 - \frac{1}{e}\right)\left(1 - \frac{4\epsilon B}{1-\epsilon^2}\right)\mathsf{opt}$.
Với nhiễu cộng, chúng ta cũng xét 2 trường hợp như vậy:
-Nếu $f(S_{i+1}) \geq \mathsf{opt}$, ta có:

$$\begin{aligned} f(S) &\geq \max\{F(S_t), F(e_m)\} - \epsilon \geq \max\{f(S_t), f(e_m)\} - 2\epsilon \\ &\geq \frac{f(S_{t+1})}{2} - 2\epsilon \\ &\geq \frac{\mathsf{opt}}{2} - 2\epsilon. \end{aligned}$$

- Nếu $f(S_{t+1}) < \mathsf{opt}$, tương tự với Bổ đề 3.1 ta có:

$$\mathsf{opt} - f(S_t) \leq \frac{B}{c(s_{t+1})}\left(f(S_{t+1}) - f(S_t)\right) + 4\epsilon B. \tag{3.7}$$

Sắp xếp lại các bất đẳng thức trên, ta có:

$$\begin{aligned} f(S_{t+1}) - (\mathsf{opt} - 4\epsilon B) &\geq (1 - \frac{c(s_{t+1})}{B})\left(f(S_t) - (\mathsf{opt} - 4\epsilon B)\right) \\ \ldots &\geq -\prod_{i=0}^{t}\left(1 - \frac{c(s_{i+1})}{B}\right)(\mathsf{opt} - 4\epsilon B). \end{aligned}$$

Suy ra:

$$
\begin{aligned}
f(S_{t+1}) &\geq \left(1 - \prod_{i=1}^{t+1}\left(1 - \frac{c(s_i)}{B}\right)\right)\mathsf{opt} + 4\epsilon B\prod_{i=1}^{t+1}\left(1 - \frac{c(s_i)}{B}\right) - 4\epsilon B \\
&\geq \left(1 - \prod_{i=1}^{t+1}\left(1 - \frac{c(s_i)}{c(S_{t+1})}\right)\right)\mathsf{opt} - 4\epsilon B \\
&\geq \left(1 - \frac{1}{e}\right)\mathsf{opt} - 4\epsilon B.
\end{aligned}
$$

Bất đẳng thức ở giữa là do biểu thức $4\epsilon B\prod_{i=1}^{t+1}\left(1 - \frac{c(s_i)}{B}\right) > 0$. Vì vậy,

$$
\begin{aligned}
f(S) &\geq \frac{f(S_{t+1})}{2} - 2\epsilon \\
&\geq \frac{1}{2}\left(1 - \frac{1}{e}\right)\mathsf{opt} - 2\epsilon(B+1).
\end{aligned}
$$

Kết hợp tất cả các trường hợp trên ta có điều phải chứng minh. □

Thuật toán này có độ phức tạp là $O(n^2)$ (về thời gian chạy và bộ nhớ cần sử dụng, vì thuật toán là tuần tự). Ta có thể cải tiến hiệu quả của **GUN** bằng cách cải tiến phương pháp đếm được đề xuất trong [131, 76] nhưng nó lại làm gia tăng độ phức tạp của thuật toán lên $O(n^5)$. Điều này làm cho thuật toán trở nên khó khăn khi áp dụng với dữ liệu lớn. Do đó, cần đề xuất các thuật toán hiệu quả hơn về không gian lưu trữ và thời gian chạy.

3.4.3. Sử dụng kỹ thuật luồng cải tiến thuật toán xấp xỉ

Để cải tiến thuật toán tham lam, luận án đề xuất một thuật toán luồng 1 lần quét (tức là duyệt tập cơ sở 1 lần) cho bài toán SMKN. **GUN** phải quét tập cơ sở nhiều lần để chọn ra phần tử cho tỉ lệ lợi nhuận biên và chi phí lớn nhất. Do đó, làm tốn thời gian chạy. Sử dụng kỹ thuật luồng, tại một thời điểm ta chỉ quét 1 lượng phần tử nhỏ so với toàn tập cơ sở và lưu trong bộ nhớ. Như vậy, thời gian và dung lượng bộ nhớ đều được giảm xuống.

Cụ thể, thuật toán thiết lập mối quan hệ giữa lời giải hiện tại đang xét với lời giải tối ưu bằng 2 kỹ thuật sau:

- Với một phần tử được quan sát, chúng ta sẽ lọc lại các phần tử có đóng góp lớn bằng cách so sánh tỉ lệ giữa hàm mục tiêu với tổng chi phí của một lời giải dự tuyển với một ngưỡng đã cho;

- Kết hợp với phương pháp thiết kế luồng do Bandanidiyuru và cộng sự đề

xuất [5] để tự động cập nhật giá trị lớn nhất của một phần tử (*maximum singleton value*) $e_m = \max_{e \in V} F(e)$.

Để mô tả dễ dàng hơn, đầu tiên ta tiếp cận phương pháp bằng cách đề xuất một phiên bản đơn giản của thuật toán. Ta giả sử rằng giá trị tối ưu opt đã biết. Tác dụng của thuật toán là để giới hạn khoảng bao của giá trị và phân tích độ phức tạp. Sau đó, phiên bản thuật toán chính sẽ loại bỏ giá trị tối ưu này đi nhưng giữ lại khoảng bao của nó.

3.4.3.1. Thuật toán xấp xỉ với giả định đã biết opt: optStr

Đầu tiên, ta tiếp cận phương pháp này bằng một phiên bản đơn giản đó là giả sử đã biết lời giải tối ưu. Khi đó thuật toán nhận v là một ước lượng của opt với $\mathsf{opt} = f(S^*)$ là giá trị tối ưu của hàm $f(\cdot)$ với lời giải tối ưu là S^*, sao cho $v \leq \mathsf{opt}$, và $\alpha \in (0, 1]$ được sử dụng để điều chỉnh điều kiện lọc đối với một phần tử được quan sát.

Sự hoạt động của thuật toán đối với trường hợp nhiễu nhân được mô tả như sau (nhiễu cộng cũng sẽ hoạt động tương tự). Thuật toán khởi tạo mới lời giải S là một tập rỗng. Với mỗi phần tử đang được xét e, thuật toán sẽ cập nhật e_m ở dòng 3 Thuật toán 8, sau đó kiểm tra điều kiện tổng chi phí sau khi thêm $c(S) + c(e)$ đã vượt B hay chưa. Nếu nó chưa vượt thì thuật toán sẽ thêm e vào trong tập S nếu:

$$\frac{F(S+e)}{c(S+e)(1-\epsilon)} \geq \frac{\alpha v}{B}.$$

Trong trường hợp ngược lại, thuật toán sẽ bỏ qua e và xét phần tử tiếp theo. Bước này sẽ giúp cho thuật toán chọn được một phần tử có đóng góp lớn nhất đồng thời loại bỏ các phần tử không tốt. Biểu thức $\frac{F(S+e)}{c(S+e)(1-\epsilon)}$ đo tỉ lệ giữa giá trị hàm mục tiêu và tổng chi phí của lời giải đang xét dưới tác động của nhiễu nhân. Sau khi kết thúc vòng lặp chính, thuật toán trả về giá trị lớn hơn giữa $\{e_m\}$ và S. Chi tiết của thuật toán được nêu ở Thuật toán 8, **optStr**.

Giả sử đặt S là lời giải dự tuyển sau khi kết thúc vòng lặp chính, để phân tích hiệu quả của thuật toán cần sử dụng các bổ đề dưới đây.

Bổ đề 3.2. *Dưới mô hình nhiễu nhân, giả sử rằng không có phần tử $e \in S^* \setminus S$ nào thỏa mãn $\frac{F(S+e)}{1-\epsilon} \geq \frac{\alpha v c(S+e)}{B}$, ta có:*

$$f(S) \geq v - \alpha v \left(\frac{1}{1-\epsilon} + \frac{(3\epsilon - \epsilon^2)}{(1-\epsilon^2)} B \right). \tag{3.8}$$

Algorithm 8 : optStr

Input: V, F, $\epsilon \in (0,1)$, B, α, v: $v \leq \mathsf{opt}$
Output: a solution S

$S \leftarrow \emptyset$
for all $e \in V$ **do**
  $e_m \leftarrow \arg\max_{e' \in \{e, e_m\}} F(e')$
  if $c(S) + c(e) \leq B$ **then**
    if F: ϵ-multiplicative noise oracle **then**
      if $\frac{F(S+e)}{c(S+e)(1-\epsilon)} \geq \frac{\alpha v}{B}$ **then**
        $S \leftarrow S + \{e\}$
      end if
    end if
    if F: ϵ-additive noise oracle **then**
      if $\frac{F(S+e)+\epsilon}{c(S+e)} \geq \frac{\alpha v}{B}$ **then**
        $S \leftarrow S + \{e\}$
      end if
    end if
  end if
end for
return $\arg\max_{S' \in \{e_m, S\}} F(S')$

Chứng minh. Từ giả thiết của bổ đề, nếu với bất kì $e \notin S$, ta sẽ có $\frac{F(S+e)}{1-\epsilon} \geq \frac{\alpha v c(S+e)}{B}$ và $c(S+e) > B$. Vì vậy:

$$\begin{aligned}
f(e|S) = f(S+e) - f(S) &\leq \frac{F(S+e)}{1-\epsilon} - \frac{F(S)}{1+\epsilon} \\
&= \frac{F(S+e)}{1-\epsilon} - \frac{1-\epsilon}{1+\epsilon}\frac{F(S)}{1-\epsilon} \\
&\leq \frac{\alpha v}{B}\left(\frac{c(S+e)}{1-\epsilon} - \frac{1-\epsilon}{1+\epsilon}c(S)\right) \quad (\text{Vì } \tfrac{F(S)}{1-\epsilon} \geq c(S)) \\
&\leq \frac{\alpha v c(e)}{B(1-\epsilon)} + \frac{\alpha v(3\epsilon - \epsilon^2)c(S)}{B(1-\epsilon^2)} \\
&\leq \frac{\alpha v c(e)}{B(1-\epsilon)} + \frac{\alpha v(3\epsilon - \epsilon^2)}{(1-\epsilon^2)}.
\end{aligned}$$

Đặt $S^* = \{s_1, s_2, \ldots, s_l\}$ là lời giải tối ưu, $S' = S^* \setminus S = \{s'_1, s'_2, \ldots, s'_j\}$, và $S'_i = \{s'_1, s'_2, \ldots, s'_i\}, i \leq j$. Bằng tính chất submodular và đơn điệu tăng của $f(\cdot)$,

ta có:

$$f(S \cup S^*) - f(S) \leq \sum_{i=1}^{j} (f(S \cup S'_i) - f(S \cup S'_{i-1}) \leq \sum_{e \in S'} f(e|S) \tag{3.9}$$

$$\leq \sum_{e \in S'} \left(\frac{\alpha v c(e)}{B(1-\epsilon)} + \frac{\alpha v (3\epsilon - \epsilon^2)}{(1-\epsilon^2)} \right) \tag{3.10}$$

$$\leq \frac{\alpha v}{1-\epsilon} + \frac{3\epsilon - \epsilon^2}{1-\epsilon^2} \alpha v B. \tag{3.11}$$

Suy ra:

$$f(S) \geq v - \alpha v \left(\frac{1}{1-\epsilon} + \frac{3\epsilon - \epsilon^2}{1-\epsilon^2} B \right).$$

Ta có điều phải chứng minh. □

Bổ đề tương tự với nhiễu cộng được xây dựng.

Bổ đề 3.3. *Dưới mô hình nhiễu cộng, giả sử không có phần tử $e \in S^* \setminus S$ nào thỏa mãn $\frac{F(S+e)+\epsilon}{c(S+e)} \geq \frac{\alpha v}{B}$, ta có: $f(S) \geq v(1-\alpha) - 2\epsilon B$.*

Chứng minh. Chứng minh tương tự với Bổ đề 3.2, với một phần tử $e \in S^* \setminus S$, ta có:

$$f(e|S) \leq F(S+e) - F(S) + 2\epsilon \leq \frac{c(e)\alpha v}{B} + 2\epsilon.$$

Vì vậy, $f(S \cup S^*) - f(S) \leq \sum_{e \in S'} f(e|S) \leq \alpha v + 2\epsilon B$. Suy ra, $f(S) \geq v(1-\alpha) - 2\epsilon B$. □

Đảm bảo hiệu quả của thuật toán **optStr** được thể hiện qua Định lý 3.2 và Định lý 3.3 được trình bày dưới đây.

Định lý 3.2. *Dưới mô hình nhiễu nhân, thuật toán* **optStr** *trả về một lời giải S_{str} thỏa mãn:*

$$f(S_{str}) \geq \min \left\{ \left(\frac{1-\epsilon}{1+\epsilon} \right)^2 \frac{\alpha v}{2}, v - \alpha v \left(\frac{1}{1-\epsilon} + \frac{3\epsilon - \epsilon^2}{1-\epsilon^2} B \right) \right\}. \tag{3.12}$$

Chứng minh. Với một phần tử $e \in S^* \setminus S$, ta có thể thấy e không thỏa mãn điều kiện nêu ở dòng 6 (hoặc 11) Thuật toán 8 hoặc e làm cho $c(S+e) > B$.
Nếu $\frac{F(S+e)}{(1-\epsilon)c(S+e)} < \frac{\alpha v}{B}$, thì định lý này sẽ thỏa mãn do Bổ đề 3.2. Vì vậy, ta sẽ xem

xét trường hợp còn lại. Do tính submodular của $f(\cdot)$, ta có:

$$f(S)+f(e) \geq f(S+e) \geq \frac{F(S+e)}{1+\epsilon} = \frac{1-\epsilon}{1+\epsilon}\frac{F(S+e)}{1-\epsilon} \geq \frac{1-\epsilon}{1+\epsilon}\frac{\alpha v c(S+e)}{B} \geq \frac{1-\epsilon}{1+\epsilon}\alpha v.$$

Suy ra, một trong 2 giá trị $f(S)$ và $f(e)$ sẽ ít nhất bằng $\frac{1-\epsilon}{1+\epsilon}\frac{\alpha v}{2}$. Suy ra:

$$f(S_{str}) \geq \frac{F(S_{str})}{1+\epsilon} \geq \max\frac{1}{1+\epsilon}\{F(S), F(e_m)\} \tag{3.13}$$

$$\geq \max\frac{1}{1+\epsilon}\{F(S), F(e_{max})\} \tag{3.14}$$

$$\geq \frac{1-\epsilon}{1+\epsilon}\max\{f(S), f(e_{max})\} \geq (\frac{1-\epsilon}{1+\epsilon})^2\frac{\alpha v}{2}. \tag{3.15}$$

Kết hợp tất cả các trường hợp trên, ta có điều phải chứng minh. □

Tương tự với Định lý 3.2, ta cũng có tỉ lệ xấp xỉ của thuật toán **optStr** dưới mô hình nhiễu cộng tuân theo định lý sau:

Định lý 3.3. *Dưới mô hình nhiễu cộng, thuật toán* **optStr** *trả về lời giải* S_{str} *thỏa mãn:*

$$f(S_{str}) \geq \min\left\{\frac{\alpha v}{2} - 3\epsilon, v(1-\alpha) - 2\epsilon B\right\}.$$

Từ Định lý 3.2 và Định lý 3.3, ta dễ dàng suy ra được hệ quả sau chỉ ra giá trị của α để thuật toán đạt được tỉ lệ xấp xỉ tốt nhất.

Hệ quả 3.1. *Tỉ lệ xấp xỉ của thuật toán* **optStr** *đạt giá trị lớn nhất là*

$$(\frac{1-\epsilon}{1+\epsilon})^2\frac{v}{(\frac{1-\epsilon}{1+\epsilon})^2 + 2(\frac{1}{1-\epsilon} + B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$$

khi $\alpha = \frac{2}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$ *với nhiễu nhân* ϵ*; và đạt giá trị lớn nhất là*

$$\frac{v}{3} + \epsilon - 2\epsilon B$$

khi $\alpha = \frac{2}{3} + \frac{2\epsilon}{v} - \frac{4\epsilon B}{3v}$ *xét với nhiễu cộng* ϵ*.*

3.4.3.2. Thuật toán xấp xỉ tổng quát: NS

Khi giải bài toán thực tế, phải bỏ đi giả thiết đã biết lời giải tối ưu. Phần này sẽ giới thiệu thuật toán luồng tổng quát, **NS** (*Noises processed Streaming*), dựa

trên các thiết lập tương tự như thuật toán **optStr**.

Đầu tiên giải quyết bài toán với nhiễu nhân. Thuật toán nhận một tham số đầu vào là $\gamma \in (0,1)$ và giá trị của α được thiết lập bằng $\alpha = \frac{2}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$ để đạt được tỉ lệ xấp xỉ tốt nhất. Đặt $e_{max} = \arg\max_{e\in V} f(e)$ và $e_m = \arg\max_{e\in V} F(e)$, ta có:

$$\frac{F(e_m)}{1+\epsilon} \leq f(e_{max}) \leq \mathsf{opt} \leq Bf(e_{max}) \leq \frac{F(e_m)}{1-\epsilon}B. \tag{3.16}$$

Các biểu thức trong (3.16) cho ta một giới hạn của giá trị tiên đoán của v của opt. Tập O sẽ gồm các giá trị tiên đoán này và được sử dụng để giới hạn các lời giải dự tuyển được xem xét. Vì vậy ta sử dụng các giá trị $v = (1+\gamma)^j \in O$ cho tập O được gán bằng:

$$O = \{(1+\gamma)^j | \frac{F(e_m)}{1+\epsilon} \leq (1+\gamma)^j \leq \frac{F(e_m)}{1-\epsilon}B, j \in \mathbb{Z}_+\}.$$

Tuy nhiên, để tìm được e_m, cần đặt ra yêu cầu rằng phải có ít nhất 1 lần quét tập V. Vì vậy, thuật toán sẽ vận dụng phương pháp cập nhật tự động được đề cập trong [5]. Trong thuật toán này, họ cập nhật $m = \max\{m, F(e)\}$ với một phần tử e đã quan sát để dự đoán miền của giá trị tối ưu sẽ nằm trong khoảng nào. Phương pháp này giúp thuật toán có thể duy trì một ước lượng tốt cho lời giải tối ưu, khi khoảng đang xét đó tăng thêm thì các phần tử tiếp theo sẽ được xem xét tiếp.

Với mỗi $v \in O$, thuật toán cập nhật lời giải S_v bằng cách thêm một phần tử mới e nếu $c(S_v + e) \leq B$ và $\frac{F(S_v+e)}{1-\epsilon} \geq \frac{\alpha v c(S_v+e)}{B}$. Sau khi kết thúc vòng lặp chính, sẽ có một giá trị $v \in O$ sao cho $(1-\gamma)\mathsf{opt} \leq v \leq \mathsf{opt}$. Cuối cùng thuật toán sẽ chọn lời giải tốt nhất giữa các lời giải dự tuyển S_v nêu trên bằng tìm kiếm nhị phân.

Với nhiễu cộng, thuật toán hoạt động tương tự, nhưng lúc này tập O sẽ là:

$$O = \{(1+\gamma)^j | F(e_m) - \epsilon \leq (1+\gamma)^j \leq (F(e_m)+\epsilon)B, j \in \mathbb{Z}_+\}.$$

Với mỗi $v \in O$ ta thiết lập $\alpha_v = \frac{2}{3} + \frac{2\epsilon}{v} - \frac{4\epsilon B}{3v}$ như đã đề cập ở phần trên. Chi tiết của thuật toán sẽ được mô tả trong trong Thuật toán 9.

Bổ đề sau xây dựng mối quan hệ giữa giá trị ước lượng v và opt:

Bổ đề 3.4. *Tồn tại một giá trị $v \in O$ thỏa mãn $(1-\gamma)\mathsf{opt} \leq v \leq \mathsf{opt}$.*

Chứng minh. Vì $\mathsf{opt} \in [\frac{F(e_m)}{1+\epsilon}, \frac{BF(e_m)}{1-\epsilon}]$, đặt $j = \lfloor \log_{1+\gamma} \mathsf{opt} \rfloor$ và nhận thấy khi $v = (1+\gamma)^j$, chúng ta có: $v = (1+\gamma)^j \leq \mathsf{opt} \leq \frac{Bm}{1-\epsilon}$ và vì vậy: $v \geq (1+\gamma)^{\log_{1+\gamma}(\mathsf{opt})-1} = \frac{\mathsf{opt}}{1+\gamma} \geq \mathsf{opt}(1-\gamma)$. □

Algorithm 9 : NS

Input: V, F,$\epsilon \in (0,1)$,B, $\gamma \in (0,1)$

Output: A solution S

$m \leftarrow 0$, $O = \{(1+\gamma)^i | i \in \mathbb{Z}_+\}$
for all $v \in O$ **do**
  $S_v = \emptyset$
end for
if f: an ϵ-multiplicative noise oracle **then**
  $\alpha = \frac{2}{\frac{1-\epsilon}{1+\epsilon}+\frac{2}{1-\epsilon}+2B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2}}$
  for all $e \in V$ **do**
    $e_m \leftarrow \arg\max_{e' \in \{e_m, e\}} F(e')$, $m \leftarrow f(e_m)$
    $O = \{(1+\gamma)^i | \frac{m}{1+\epsilon} \leq (1+\gamma)^i \leq \frac{Bm}{1-\epsilon}, i \in \mathbb{Z}_+\}$
    Delete S_v for each $v \notin O$
    for all $v \in O$ and $c(S_v + e) \leq B$ **do**
      if $\frac{F(S_v+e)}{1-\epsilon} \geq \frac{\alpha v c(S_v+e)}{B}$ **then**
        $S_v \leftarrow S_v + \{e\}$
      end if
    end for
  end for
else if f: ϵ-additive noise oracle **then**
  for all $e \in V$ **do**
    $e_m \leftarrow \arg\max_{e' \in \{e_m, e\}} F(e')$, $m \leftarrow f(e_m)$
    $O = \{(1+\gamma)^i | m - \epsilon \leq (1+\gamma)^i \leq B(m+\epsilon), i \in \mathbb{Z}_+\}$
    Delete S_v for each $v \notin O$
    for all $v \in O$ and $c(S_v + e) \leq B$ **do**
      $\alpha_v \leftarrow \frac{2}{3} + \frac{2\epsilon}{v} - \frac{4\epsilon B}{3v}$
      if $F(S_v + e) + \epsilon \geq \frac{\alpha v c(S_v+e)}{B}$ **then**
        $S_v \leftarrow S_v + \{e\}$
      end if
    end for
  end for
end if
// Binary Search on S_v
Find $S_{str} \leftarrow \arg\max_{S \in \{S_v : v \in O\} \cup \{e_m\}} F(S_v)$ by BS
return S_{str}

Từ đó xây dựng được định lý phân tích đảm bảo lý thuyết sau:

Định lý 3.4. *Với nhiễu nhân ϵ, thuật toán* **NS** *là thuật toán luồng 1 lần quét cho độ phức tạp truy vấn $O(\frac{n}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$, độ phức tạp bộ nhớ $O(\frac{B}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$ và có tỉ lệ xấp xỉ là $(\frac{1-\epsilon}{1+\epsilon})^2\frac{(1-\gamma)}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$.*

Chứng minh. Thuật toán có độ phức tạp truy vấn là $O(\frac{n}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$ vì có nhiều nhất $\frac{1}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon})$ phần tử trong O và mỗi lời giải dự tuyển S_v cần $O(n)$ truy vấn. Mặt khác, mỗi S_v có nhiều nhất B phần tử, không gian lưu trữ là $O(\frac{B}{\gamma}\log(B\frac{1+\epsilon}{1-\epsilon}))$.

Từ Bổ đề 3.4, sẽ có một $v \in O$ sao cho $(1-\gamma)\mathsf{opt} \leq v \leq \mathsf{opt}$. Áp dụng Định lý 3.2 và thiết lập giá trị của α trong thuật toán, có được:

$$f(S_{str}) \geq (\frac{1-\epsilon}{1+\epsilon})^2\frac{v}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})} \geq (\frac{1-\epsilon}{1+\epsilon})^2\frac{(1-\gamma)\mathsf{opt}}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}.$$

Ta có điều phải chứng minh. □

Tương tự với Định lý 3.4, đảm bảo chất lượng lời giải của Thuật toán 9 được khẳng định bằng định lý sau:

Định lý 3.5. *Dưới tác động của nhiễu cộng ϵ, thuật toán* **NS** *là thuật toán luồng 1 lần quét với độ phức tạp truy vấn là $O(\frac{n}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon}))$, độ phức tạp bộ nhớ là $O(\frac{B}{\gamma}\log(\frac{mB+\epsilon}{m-\epsilon}))$ và trả về kết quả S_{str} thỏa mãn $f(S_{str}) \geq \frac{1-\gamma}{3}\mathsf{opt} + \epsilon - 2\epsilon B$, với $m = \max_{e\in V} F(e)$.*

3.5. Nghiên cứu thực nghiệm

Về mặt lý thuyết, các thuật toán luồng được đề xuất sẽ tối ưu về mặt thời gian, số lượng bộ nhớ cần lưu trữ và số lượng câu truy vấn cần dùng để gọi tới hàm F. Để cụ thể hóa, luận án đưa ra một số thực nghiệm để so sánh các thuật toán đã đề xuất cho một trường hợp cụ thể của SMKN, đó là xét ước lượng nhiễu trên *tối đa ảnh hưởng với ràng buộc chi phí (Influence Maximization under Knapsack constraint- IMK).*

3.5.1. Ứng dụng IMK dùng cho thực nghiệm

IM là một bài toán quan trọng trong nghiên cứu về ảnh hưởng xã hội và lan truyền tiếp thị [43, 133, 132, 71, 30]. Mục tiêu của bài toán này là tìm một tập k người dùng có thể ảnh hưởng tới các người dùng khác trong mạng xã hội là lớn nhất. Lúc này, bài toán là tối đa hàm submodular với ràng buộc lực lượng. Tuy nhiên, ta cũng có thể coi k này là ngân sách tối đa để chọn ra tập người dùng làm

hạt giống trong việc lan truyền thông tin. Như vậy, chúng ta có thể xem xét bài toán IM với ràng buộc ngân sách để giải bài toán tổng quát khi mà thực tế rằng mỗi người dùng khác nhau cần có một chi phí khác nhau để bắt đầu gây ảnh hưởng đến họ [105]. Bài toán trở thành *tối đa ảnh hưởng với ràng buộc chi phí - IMK*.

Để giải bài toán này cần thực hiện các bước xây dựng mô hình, xây dựng thuật toán để giải. Mô hình được xây dựng dựa trên mô hình lan truyền thông tin Bậc độc lập, IC, được đề xuất bởi Kempe và cộng sự [43]. Sau đó luận án sẽ đưa ra định nghĩa bài toán IM với ràng buộc chi phí dựa trên mô hình này.

a) Mô hình lan truyền thông tin IC

Cho đồ thị $G = (V, E)$ biểu diễn một mạng xã hội trong đó tập đỉnh V biểu diễn người dùng và các cạnh E là các liên kết xã hội (kết bạn, like, comment...). Mỗi cạnh (u, v) có một xác suất $p(u, v) \in [0, 1]$ để biểu diễn sự truyền tải thông tin từ u đến v. Trong V, một tập hạt giống hay còn gọi là tập người gây ảnh hưởng (*seed set*), $S \subseteq V$, tiến trình ảnh hưởng dưới mô hình Bậc độc lập [43], IC, xảy ra theo các bước rời rạc như sau:

- Tại bước $t = 0$, mọi nút trong S đều bị ảnh hưởng; Những nút bị ảnh hưởng gọi là nút được kích hoạt; Những nút chưa bị ảnh hưởng gọi là chưa kích hoạt;

- Tại bước $t \geq 1$, mỗi nút u đã bị ảnh hưởng tại bước $t - 1$ sẽ chỉ có một cơ hội duy nhất để ảnh hưởng tới nút hàng xóm v mà chưa bị ảnh hưởng của nó với một xác suất thành công là $p(u, v)$. Nếu một nút đã bị ảnh hưởng, thì nó sẽ được duy trì trạng thái này cho tới kết thúc quá trình ảnh hưởng.

- Tiến trình ảnh hưởng này sẽ kết thúc tại bước thứ t nếu không có một nút mới nào được kích hoạt tại bước này.

b) Bài toán IMK

Đặt $f : 2^V \mapsto \mathbb{R}_+$ là một hàm ảnh hưởng của một tập hạt giống S nào đó. Đây là hàm số người ảnh hưởng mong muốn sau khi quá trình ảnh hưởng thông tin xảy ra. Kempe [43] đã chỉ ra rằng f là đơn điệu và submodular khi lan truyền thông tin dưới mô hình IC. Giả sử rằng mỗi nút u có một chi phí $c(u) > 0$ biểu thị về sự "cố gắng bao nhiêu" hoặc "cần tốn bao nhiêu" hoặc "cần cố bao nhiêu" để đỉnh đó sẽ bắt đầu gây ảnh hưởng tới những đỉnh khác. Cho một ngân sách B, bài toán yêu cầu cần tìm tập hạt giống S với tổng chi phí $\sum_{u \in S} \leq B$ để $f(S)$ là cực đại. Hiển nhiên, để đỉnh u đó gây ảnh hưởng tới các đỉnh khác, thì u cũng được gọi là một đỉnh bị ảnh hưởng.

Mặc dù có rất nhiều phương pháp giải bài toán IMK [105, 104], nhưng nhìn chung đều dựa trên phương pháp *Lấy mẫu ảnh hưởng ngược (Reachable Influence*

Sampling - RIS) do Borg và cộng sự [16] đề xuất, để đạt được một ước lượng F cho f với F cũng là đơn điệu và submodular.

Ngược lại, các thuật toán xấp xỉ trong luận án là các thuật toán đầu tiên giải cho IMK với hàm ước lượng xấp xỉ có thể không còn là submodular.

c) Ước lượng nhiễu của hàm mục tiêu

Vì tính toán hàm f là #P-khó dưới mô hình IC [33], nên ta tìm hàm F là một ước lượng nhiễu ϵ của f. Ta sẽ áp dụng *phương pháp tính dựa trên lưới (sketch-based method)* của Cohen [36].

Cho $k \in \mathbb{Z}_+$, ước lượng F được xây dựng như sau: Mỗi nút v được gán một cặp "node-instance" $(v, i) \in V \times \{1, .., l\}$ trong đó $l \geq 1$ là số thể hiện biểu thị các dạng của đồ thị G được biểu diễn trên lưới. Mỗi cặp (v, i) có một giá trị xếp hạng (rank) độc lập $r_v^i \sim \cup[0, 1]$ với $\cup[0, 1]$ là phân phối chuẩn trong miền $[0, 1]$. Với mọi đỉnh $v \in V$, các lưới gọi là *lưới liên kết có thể đến được (combined reachability sketches)* X_v là các giá trị k nhỏ nhất trong $\{r_v^i | (v, i) \in R_u\}$ với R_u bao gồm các nút v có thể đến được từ một nút u nằm trong $G^{(i)}$.

Như đã được đề cập đến trong nghiên cứu của Kempe [43], G là một thể hiện của đồ thị được sinh ra trong mô hình ảnh hưởng. Vì vậy, sử dụng $\{G^{(i)}\}$ để biểu diễn tập các thể hiện lan truyền thông tin được sinh ra bởi lấy mẫu Monte Carlo tùy thuộc vào mô hình ảnh hưởng.

Cohen [36] đã chỉ ra rằng ảnh hưởng của tập S đối với các thể hiện $\{G^{(i)}\}$ có thể ước lượng theo công thức:

$$Inf(\{G^{(i)}\}, S) = \frac{1}{l} \sum_{i \in l} |\bigcup_{u \in S} R(G^{(i)}, u)| = \frac{1}{l} |\bigcup_{u \in S} R_u|.$$

Vì vậy kích thước của tập hợp $|\bigcup_{u \in S} R_u|$ sẽ được ước lượng bởi lưới liên kết X_u với $u \in S$. Bên cạnh đó, hàm ước lượng là $F(X)$ với $X \subseteq S$. Sử dụng xếp hạng theo ngưỡng (threshold rank), τ_u, cho mỗi nút u sao cho: $\tau_u = k^{th}(r_v^i | (v, i) \in R_u)$ là k giá trị xếp hạng nhỏ nhất trong R_u. Cohen [36] cũng chỉ ra rằng khi $|X_u| = k$, hàm ước lượng sẽ là: $F(X) = (k - 1)/(l\tau_u)$ và $F(X) = |\cup_{u \in X}|/l$ trong trường hợp ngược lại (có nghĩa là $|X_u| < k$ với $\tau_u = 1$).

Với một hằng số $c > 0$, $k = ce^{-2}\log(n)$, các tác giả đã chứng minh được F là ước lượng nhiễu nhân ϵ của f với xác suất ít nhất là $1 - 1/n^{c-2}$. Bên cạnh đó, Crawford [**?**] cũng đã chứng minh rằng F có thể không còn là submodular. Do đó, ước lượng F của Cohen đề xuất có thể ước lượng nhiễu ϵ cho hàm f. Vì vậy, phần thực nghiệm của luận án đã sử dụng phương pháp của Cohen để tính F.

3.5.2. Thiết lập cho thực nghiệm

Bài toán SMKN là một bài toán mới nên trong thực nghiệm ta sẽ so sánh hiệu quả của hai thuật toán GUN với NS này. Để so sánh, phần thực nghiệm đã tiến hành các phép đo quan trọng về: ước lượng hàm ảnh hưởng, thời gian chạy, số lượng truy vấn và bộ nhớ.

3.5.2.1. Tập dữ liệu

Phần thực nghiệm được tiến hành trên 2 bộ dữ liệu về mạng xã hội là Facebook [136] và mạng lưới tham khảo HEPT [71] là những bộ dữ liệu thường được sử dụng trong các thực nghiệm trong các công bố về hàm submodular, như là tối đa lợi ích [105], lan truyền tiếp thị trên mạng lưới có quy mô hàng tỉ cạnh [71] hoặc xấp xỉ hàm submodular [40]. Bộ dữ liệu gồm bộ nhỏ hơn, Facebook, đã gồm hơn 1 nghìn đỉnh (1.4K) và gần 60K cạnh. HEPT là bộ dữ liệu lớn hơn với 15K đỉnh và 59K cạnh.

Bảng 3.3 đưa ra các mô tả tóm tắt về các tập dữ liệu này.

Bảng 3.3: Thống kê dữ liệu

Tập dữ liệu	Số đỉnh	Số cạnh	Loại	Bậc trung bình
Facebook [93]	1.4K	59.6K	Có hướng	82
HEPT [32]	15K	59K	Có hướng	4.1

3.5.2.2. Các thiết lập tham số

Với bài toán này, sẽ có một vài thông số thực sự ảnh hưởng tới chất lượng và thời gian tìm ra lời giải là: ngân sách B, chi phí $c(e)$ của mỗi đỉnh e trong V; k, l và c để dựng lên các lưới; và hệ số ϵ của mô hình nhiễu. Mô phỏng theo thực nghiệm của tác giả [40], F là một xấp xỉ ϵ của f bằng cách chọn k hợp lý: Với $c > 0$, $k = ce^{-2}\log(n)$. Tác giả đã thiết lập $l = 128$ để dựng các lưới và tính toán hàm F. Trong phần thực nghiệm, thiết lập $\epsilon = 0.5$ (giống một trường hợp thực nghiệm của [40]), giá trị của γ cũng được biến đổi để thể hiện vai trò của nó và đặt $c = 3$ nên ta có $k = 115$ khi thực nghiệm với NS.

Bên cạnh đó, kích thước của tập dữ liệu đầu vào cũng sẽ ảnh hưởng tới thời gian chạy và bộ nhớ cần thiết. Từ các quan sát, tác giả đã tạo ra các thực nghiệm với cấu hình: Thay đổi các giá trị của B và γ đối với cả 2 bộ dữ liệu Facebook và HEPT. Vì vậy, phần thực nghiệm được tiến hành 2 trường hợp: Đầu tiên, thử với bộ dữ liệu nhỏ, Facebook, với B từ 100 tới 500 và γ được gán lần lượt từ 0.1 tới 0.5. Thứ hai, thực nghiệm với bộ dữ liệu lớn hơn, HEPT, với B được gán lần lượt

là 30, 50, 70 và 100; γ là 0.1, 0.2 và 0.3.

Để tiện biểu diễn và kí hiệu trên hình vẽ, ta quy ước các lần chạy **NS** khi γ thay đổi từ 0.1, 0.2, 0.3, 0.4 đến 0.5 là $NS.\gamma = 0.1$, $NS.\gamma = 0.2$, $NS.\gamma = 0.3$, $NS.\gamma = 0.4$ và $NS.\gamma = 0.5$. Các kí hiệu K, M lần lượt viết tắt cho hàng nghìn và hàng triệu. Luận án cũng sử dụng phương pháp mô phỏng Monte Carlo để đạt được ước lượng cho hàm ảnh hưởng f với 10,000 lần mô phỏng [43].

Các thực nghiệm được chạy trên hệ điều hành Linux với cấu hình 2 $\times$ Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz và 4 $\times$ 16 GB DIMM ECC DDR4 @ 2400MHz.

3.5.3. Nhận xét kết quả thực nghiệm

Phần kết quả thực nghiệm cho thấy về chất lượng lời giải **GUN** cho kết quả tốt nhất, nhưng **NS** thấp hơn không đáng kể. Ngược lại, **NS** lại cho thấy sự vượt trội khi giảm số lượng truy vấn, thời gian chạy và bộ nhớ dùng cho tập dữ liệu khi chạy thuật toán (Hình 3.1 đến Hình 3.3).

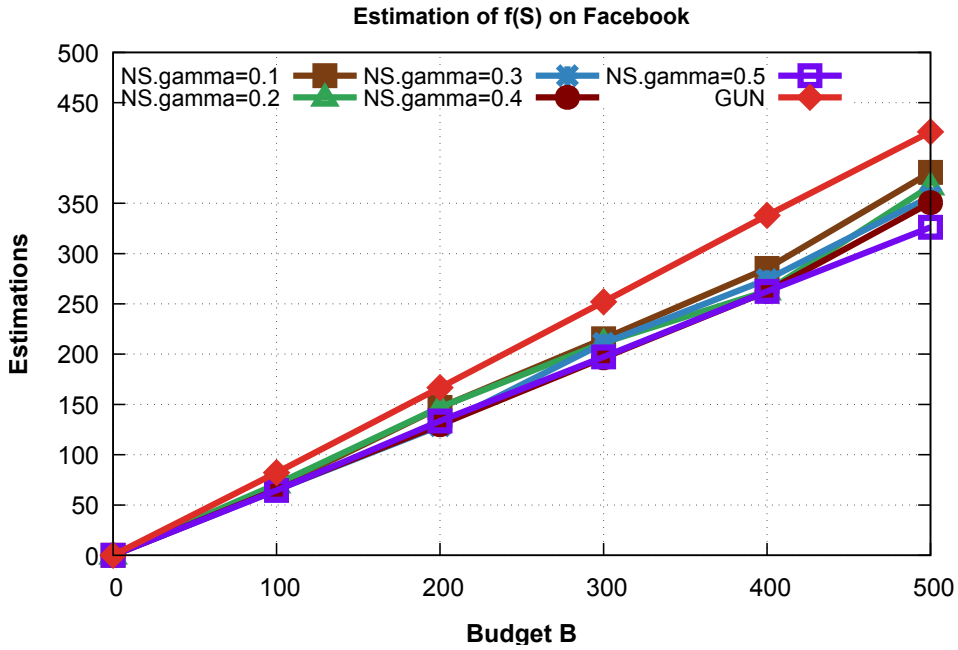
a) Ước lượng hàm ảnh hưởng

Hình 3.1 minh họa các giá trị của hàm F, một ước lượng của hàm f phụ thuộc các giá trị của B. Với hình 3.1(a), đường màu đỏ biểu diễn cho **GUN**, nằm cao hơn so với các đường khác tại tất cả các mốc B trong khi các đường của **NS** gần như là giống nhau. Cho dù các giá trị F của **GUN** cao hơn **NS**, thì sự cao hơn này cũng không đáng kể.

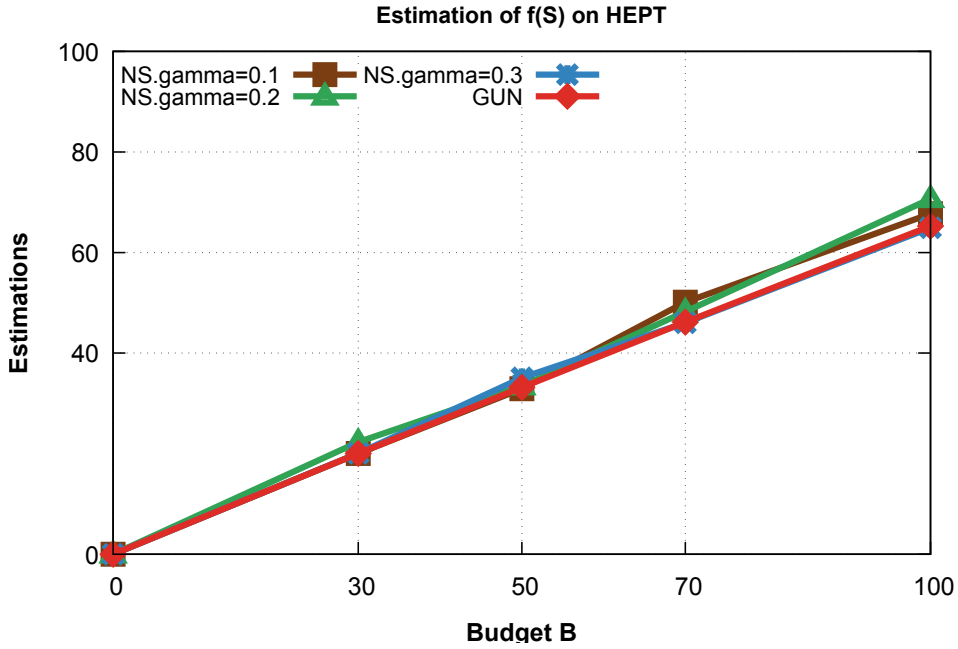
Lý do giá trị F của **GUN** cao hơn **NS** vì **GUN** chọn tất cả các e thỏa mãn điều kiện trong khi **NS** chỉ giữ lại những e nào vừa thỏa mãn điều kiện mà vừa vượt qua được ngưỡng. Tuy nhiên, các giá trị nhỏ hơn của **NS** là chấp nhận được vì nó giảm thời gian chạy và số lần truy vấn lời gọi hàm. Hình 3.1(b) chỉ ra sự hội tụ của những đường thẳng này. Tại mỗi mốc của B, giá trị F của **GUN** không cao hơn là bao so với của **NS**. Những kết quả này chỉ ra rằng khi giá trị của B hoặc kích cỡ đầu vào tăng lên , **NS** sẽ cho các ước lượng của hàm f không hề kém hơn so với **GUN** hay thậm chí tốt hơn. Đây là một kết quả quan trọng bởi vì **NS** cần phải cân bằng giữa chất lượng lời giải và thời gian chạy, đặc biệt khi dữ liệu đầu vào có kích thước lớn.

b) Thời gian chạy

Hình 3.2(a)(b) so sánh thời gian chạy theo giây của **GUN** và **NS** theo sự thay đổi của B và γ. Hình 3.2(a) minh họa 5 giá trị của γ với **NS**, 5 giá trị của B từ 100 đến 500 trong khi Hình 3.2(b) minh họa 3 giá trị của γ từ 0,1 đến 0,3 và 4 giá trị khác của B từ 30 đến 100. Hình minh họa cho thấy lần chạy **GUN** thực sự



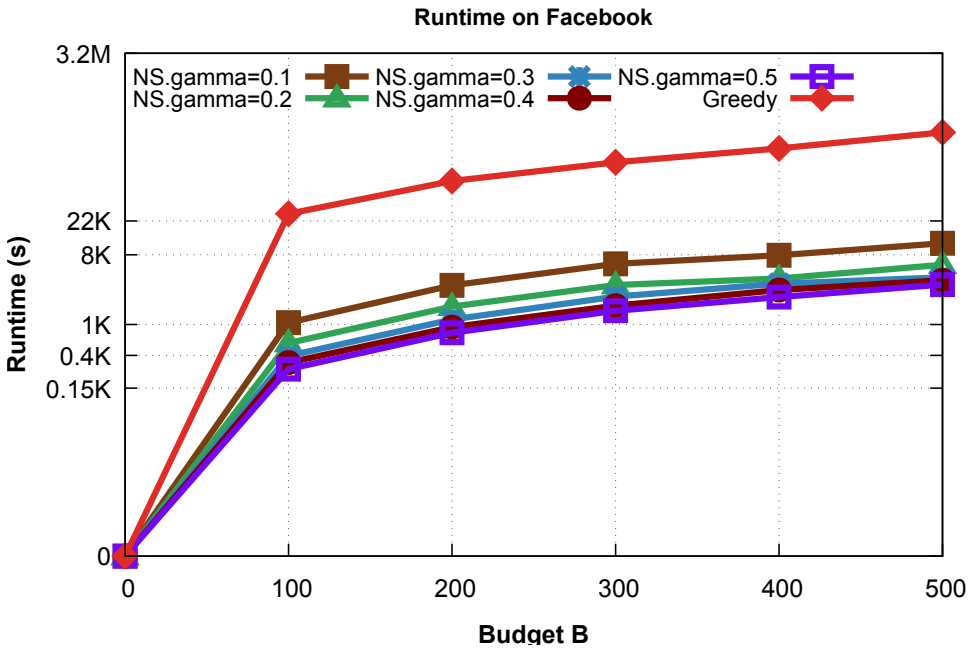
(a)



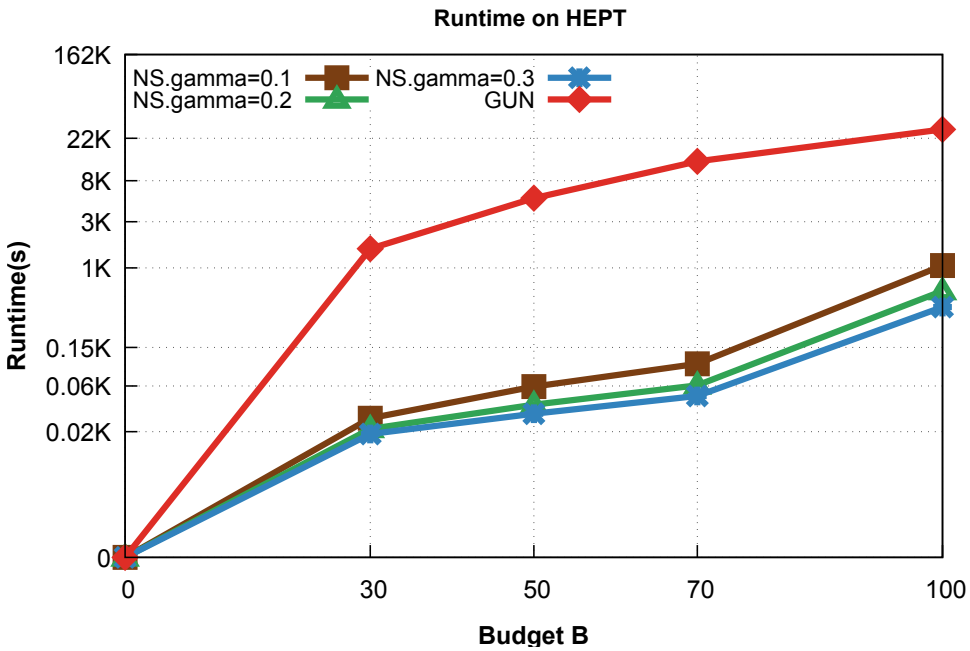
(b)

Hình 3.1: Giá trị ước lượng của hàm f với các mốc B trên 2 bộ dữ liệu Facebook và HEPT.

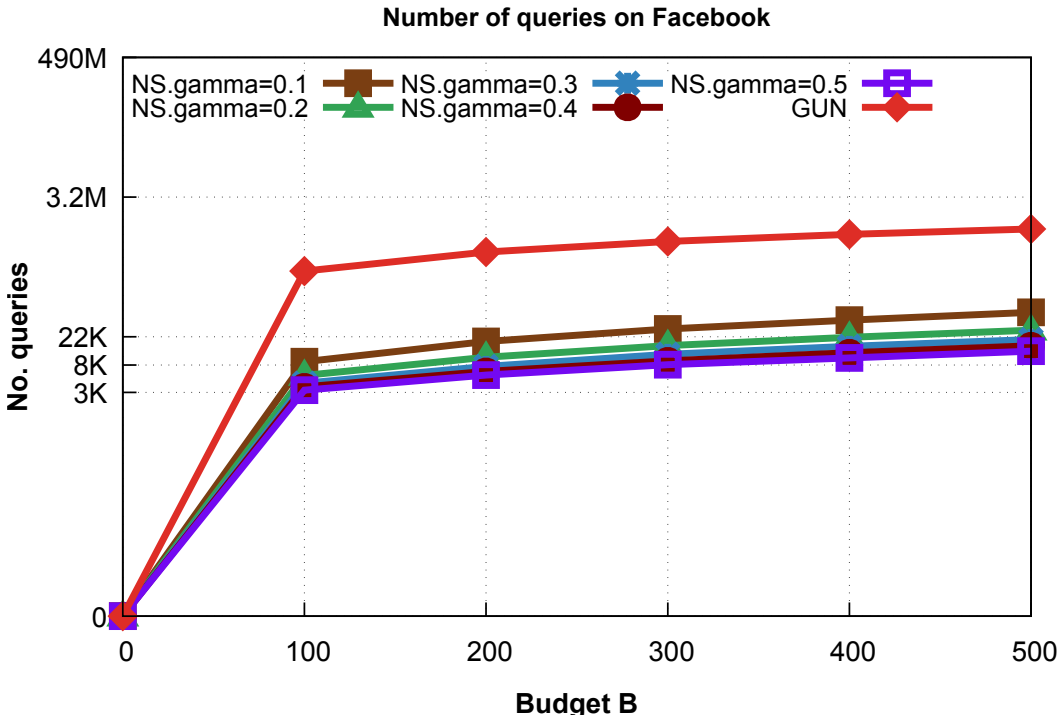
tốn nhiều thời gian hơn các lần chạy khác. Giá trị thấp hơn của γ cũng khiến mất nhiều thời gian hơn để chạy **NS**, tuy nhiên, sự khác biệt của các trường hợp này là không đáng kể. Mặt khác, thời gian chạy của **GUN** xấp xỉ cao hơn **NS** vài lần khi giá trị của B hoặc kích thước của dữ liệu đầu vào tăng lên. Điều này chứng tỏ việc áp dụng thuật toán luồng cho dữ liệu lớn tỏ ra vượt trội hơn so với tham lam.


Runtime on Facebook
3.2M
22K
8K
1K
0.4K
0.15K
0
Runtime (s)
NS.gamma=0.1
NS.gamma=0.2
NS.gamma=0.3
NS.gamma=0.4
NS.gamma=0.5
Greedy
0
100
200
300
400
500
Budget B

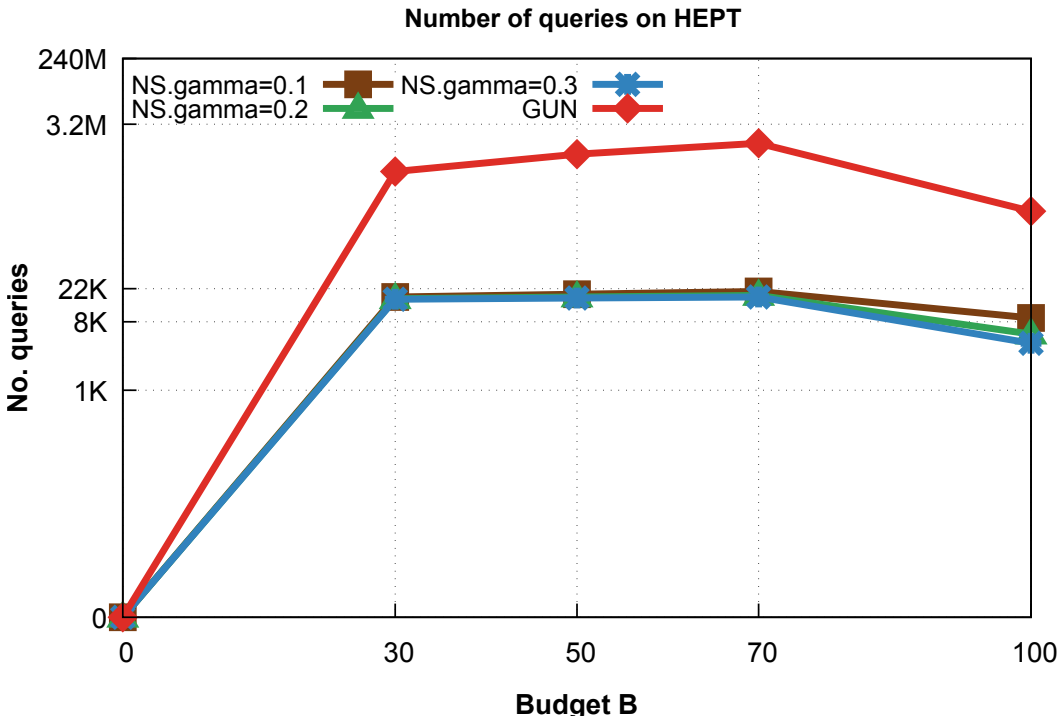
(a)


Runtime on HEPT
162K
22K
8K
3K
1K
0.15K
0.06K
0.02K
0
Runtime(s)
NS.gamma=0.1
NS.gamma=0.2
NS.gamma=0.3
GUN
0
30
50
70
100
Budget B

(b)



(c)



(d)

Hình 3.2: Thời gian chạy (a, b) và số lượng truy vấn (c, d) với các mốc B khác nhau trên 2 bộ dữ liệu Facebook và HEPT

c) Số truy vấn

Hình 3.2(c)(d) hiển thị hàng nghìn truy vấn mà cả hai thuật toán cần để tìm ra lời giải S. Kết quả chỉ ra rằng **NS** vượt trội đáng kể so với **GUN** trong mọi trường hợp của B trên cả Facebook và HEPT. Trong quá trình thiết lập B từ 100 đến 500 hoặc từ 30 đến 100 cho mỗi bộ dữ liệu và γ từ 0.1 đến 0.5, **GUN** phải sử dụng hơn 87K cho đến gần 1.044M truy vấn để thu thập các phần tử cho S. Ngược lại, **NS** với $\gamma = 0.5$ giảm xuống $\gamma = 0.1$ chỉ dành khoảng hơn 3K truy vấn đến không quá 53K, trong đó $NS.\gamma = 0.1$ cần nhiều truy vấn hơn đáng kể so với khác γ vì số lượng truy vấn của nó đã tăng từ khoảng 9K lên khoảng 53K theo sự tăng trưởng của ngân sách B.

Kết quả trên đây rất có ý nghĩa khi số lượng truy vấn cần thiết trong **NS** *thấp hơn đáng kể từ 10 đến 20 lần* so với **GUN**. Ngoài ra, khi kích thước của dữ liệu đầu vào tăng lên đối với HEPT, sự chênh lệch số lượng truy vấn của **GUN** và **NS** trở nên rõ ràng hơn nhiều. **GUN** *cao hơn 82 lần cho tới 266 lần* so với $NS.\gamma = 0.3$ và *cao hơn 72 cho tới 192 lần* so với $NS.\gamma = 0.1$. Kết luận là khi đầu vào tăng lên, số lượng truy vấn trong **GUN** cũng nhanh chóng tăng lên và **GUN** cần nhiều truy vấn hơn **NS** để tìm ra lời giải.

NS cần ít truy vấn hơn **GUN** vì **NS** loại bỏ tất cả các phần tử e không thể duy trì điều kiện của tập hợp O (dòng 9 và dòng 20 trong Thuật toán 9). Ngoài ra, điều kiện ở các dòng 12 và dòng 24 trong Thuật toán 9 cũng làm giảm không gian tìm kiếm. Do đó, số lượng truy vấn giảm mạnh. Điều này cũng giúp thời gian chạy của **NS** cực kỳ nhanh hơn **GUN**. Nhìn chung, ít truy vấn hơn chứng minh rằng thuật toán **NS** có hiệu suất cao hơn **GUN**.

d) Kích thước của các tập O khác nhau

Bảng 3.4: Sử dụng bộ nhớ (MB) của các thuật toán trên Facebook

Thuật toán	**Ngân sách B**				
	100	**200**	**300**	**400**	**500**
NS.$\gamma = 0.1$	22732	**22572**	**22588**	22588	22968
NS.$\gamma = 0.2$	22600	22732	22600	22600	**22856**
NS.$\gamma = 0.3$	**22588**	**22572**	**22588**	**22584**	22860
NS.$\gamma = 0.4$	22732	22716	22716	22732	22864
NS.$\gamma = 0.5$	22600	22732	22728	**22584**	22872
GUN	22600	22716	22600	22728	22860

Bảng 3.5: Sử dụng bộ nhớ (MB) của các thuật toán trên HEPT

Thuật toán	**Ngân sách B**			
	30	**50**	**70**	**100**
NS.$\gamma = 0.1$	55864	40524	40464	**40396**
NS.$\gamma = 0.2$	40140	56376	56312	56316
NS.$\gamma = 0.3$	56440	**40080**	40296	55740
GUN	**37432**	56056	**40076**	55864

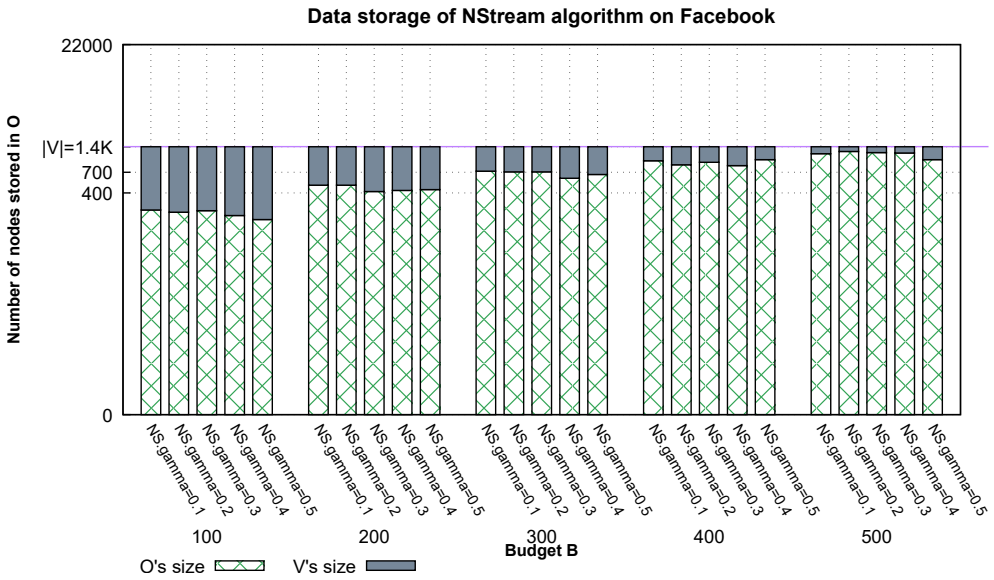
Bảng 3.4 và 3.5 cho biết mức sử dụng bộ nhớ trung bình của bộ nhớ vật lý và bộ nhớ ảo, khi **GUN** và **NS** chạy với B và γ khác nhau. Với mỗi cột B, giá trị tối thiểu được in đậm. Kết quả cho thấy sự khác biệt về bộ nhớ được sử dụng bởi chính các thuật toán là không đáng kể. Tuy nhiên, nếu chúng ta phân tích lượng dữ liệu được lưu trữ cho mỗi thuật toán, sự khác biệt là rất rõ ràng.

Thay vì tìm kiếm các phần tử trong toàn bộ tập hợp V như **GUN** nhiều lần, **NS** chỉ xem xét các phần tử trong tập hợp O bị giới hạn ở $O(\frac{B}{\gamma}\ log(B\frac{1+\epsilon}{1-\epsilon})))$ độ phức tạp bộ nhớ. Điều này giúp **NS** hoạt động tốt với dữ liệu có kích thước lớn.

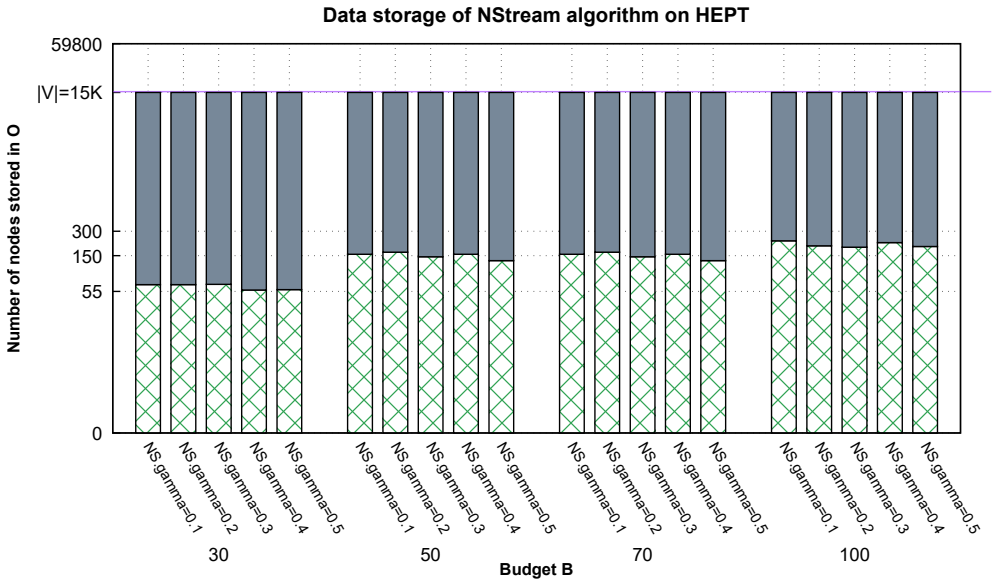
Để kiểm tra hiệu suất của **NS** khi thay đổi tham số đầu vào, **NS** được chạy với nhiều giá trị B và γ sau đó thu thập số nút tối đa được lưu trong tập hợp O để tìm ra giải pháp S. Sau đó, luận án so sánh những con số này với kích thước của tập hợp cơ sở V để cho biết **NS** tiết kiệm bộ nhớ như thế nào (Hình 3.3).

Hình 3.3 so sánh số nút được lưu trong tập hợp O và kích thước của tập hợp V cho từng trường hợp γ và B. Hình 3.3(a) vẽ kết quả của dữ liệu Facebook với γ từ 0.1 đến 0.5 và B từ 100 đến 500 và Hình 3.3(b) minh họa kết quả của dữ liệu HEPT với γ cũng từ 0.1 đến 0.5 và B trong $\{30, 50, 70, 100\}$. Ta có thể thấy rằng bộ nhớ cần thiết cho O luôn thấp hơn bộ nhớ của V. Với $B = 100$ trên Facebook, sự chênh lệch giữa bộ nhớ của O và bộ nhớ của V là lớn nhất. Tương tự, sự khác biệt về bộ nhớ giữa O và V trên HEPT là lớn nhất tại $B = 30$. Đặc biệt, một tập dữ liệu lớn như HEPT với 15K nút, bộ nhớ của O đặc biệt thấp hơn bộ nhớ của V. Ngoài ra, giá trị của B cũng ảnh hưởng tới sự chênh lệch giữa O và V. Hình 3.3(b) với giá trị B nhỏ và giá trị V lớn biểu thị sự giảm kích thước của O.

Thêm vào đó, việc thay đổi γ cũng có ảnh hưởng tới O. $\gamma = 0.5$ luôn làm cho các nút được lưu trữ của O nhỏ nhất trong khi các nút khác làm cho kích thước thay đổi không nhất quán. Đó là bởi vì O trong **NS** không chỉ phụ thuộc vào B hay γ mà còn phụ thuộc vào việc lựa chọn e giúp tối đa hóa $F(e)$. Tuy nhiên, sự khác biệt về kích thước của O với các γ khác nhau tại cùng một B là không đáng



(a)



(b)

Hình 3.3: Kích thước của O với các B và γ khác nhau trên 2 bộ dữ liệu Facebook và HEPT

kể. Nhìn chung, thử nghiệm cho thấy NS tiết kiệm đáng kể dung lượng lưu trữ dữ liệu so với GUN. Do đó, luồng là một giải pháp phù hợp cho vấn đề dữ liệu lớn.

3.6. Kết luận chương

Trong chương này, luận án đã trình bày hai thuật toán với sự đảm bảo về mặt lý thuyết để giải quyết vấn đề tối đa hóa hàm submodular với ràng buộc chi phí khi có nhiễu (nhiễu cộng hoặc nhiễu nhân). Thuật toán được đề xuất đầu tiên là

GUN cung cấp tỉ lệ xấp xỉ

$$\frac{1}{2}\frac{1-\epsilon}{1+\epsilon}\left(1-\frac{1}{e}\right)\left(1-\frac{4\epsilon B}{1-\epsilon^2}\right)$$

khi có nhiễu nhân ϵ, và trả về nghiệm S thỏa mãn

$$f(S) \geq \frac{1}{2}\left(1-\frac{1}{e}\right) f(S^*) - 2\epsilon(B+1)$$

khi có nhiễu cộng ϵ.

Tuy nhiên, thuật toán tham lam **GUN** mất rất nhiều thời gian chạy khi dữ liệu tăng lên về kích cỡ. Do đó, luận án đã cung cấp một thuật toán luồng 1 lần quét, **NS**, với các đảm bảo về mặt lý thuyết. Cụ thể, khi có nhiễu nhân ϵ, **NS** trả về tỉ lệ xấp xỉ là $(\frac{1-\epsilon}{1+\epsilon})^2\frac{(1-\gamma)}{(\frac{1-\epsilon}{1+\epsilon})^2+2(\frac{1}{1-\epsilon}+B\frac{3\epsilon-\epsilon^2}{1-\epsilon^2})}$. Trong nhiễu cộng ϵ, **NS** trả về một nghiệm S_{str} thỏa mãn $f(S_{str}) \geq \frac{1-\gamma}{3}f(S^*) + \epsilon - 2\epsilon B$. Kết quả thử nghiệm cho thấy rằng **NS** không chỉ cho hàm mục tiêu gần với **GUN** mà còn nhanh hơn nhiều lần so với **GUN**. Điều này làm cơ sở để các nghiên cứu tiếp theo ta có thể vận dụng thuật toán luồng để giải quyết vào nhiều bài toán khác cũng như áp dụng xử lý với nhiễu trong các bài toán tối ưu hàm submodular khác.

Các nghiên cứu trong chương này đã được công bố trong bài báo *Submodular Maximization Subject to a Knapsack Constraint Under Noise Models*, tạp chí Asia-Pacific Journal of Operational Research, tập 39, số 6, (2022) (**ISI/Q3**).

CHƯƠNG 4
BÀI TOÁN PHỦ SUBMODULAR ĐƠN ĐIỆU TRÊN LƯỚI NGUYÊN

Chương 2 luận án đã nghiên cứu vấn đề mở rộng hàm submodular thành k-submodular đáp ứng một số tình huống khi cần phân lớp các phần tử thành các tập không giao nhau, phục vụ một số tình huống thực tiễn như tối đa ảnh hưởng k-chủ đề, đặt k loại cảm biến... Tuy nhiên, trong thực tế còn tồn tại các tình huống một phần tử tốt *có thể được lựa chọn nhiều lần* vào tập lời giải. Ví dụ khi một công ty chọn các đại lý tốt để tiếp thị sản phẩm, công ty sẽ có xu hướng chọn đi chọn lại các đại lý mang lại lợi nhuận cao. Các tình huống như vậy cần mở rộng hàm submodular trên lưới nguyên.

Luận án đặt vấn đề nghiên cứu bài toán Phủ Submodular (Submodular Cover - SC) đơn điệu trên lưới nguyên do tính mới của bài toán này. Các nghiên cứu hiện nay còn một số hạn chế khi hàm mục tiêu cho giá trị nguyên hoặc độ phức tạp truy vấn cao. Luận án giải quyết bài toán bằng thuật toán xấp xỉ tiêu chí kép được thiết kế song song với độ phức tạp truy vấn và độ phức tạp song song thấp, khắc phục các hạn chế của các nghiên cứu hiện có.

Đối với bài toán này, luận án đưa ra các kết quả nghiên cứu lý thuyết với các đóng góp về chất lượng lời giải, độ phức tạp song song và độ phức tạp truy vấn, được chứng minh chặt chẽ qua các Bổ đề và Định lý. Bài toán SC trên lưới nguyên là hướng nghiên cứu mới, các công bố chủ yếu hiện nay là đưa ra các đảm bảo lý thuyết thuần túy.

4.1. Phát biểu bài toán, hàm mục tiêu và một số quy ước quan trọng

4.1.1. Phát biểu bài toán

Hiện nay có nhiều bài toán thực tiễn đang thu hút nhiều sự quan tâm như tối ưu ngân sách, tối ưu lực lượng với yêu cầu lợi ích thu được tối thiểu đạt mức nào đó là chấp nhận được. Các nghiên cứu như vậy, có thể áp dụng trong lan truyền tiếp thị [63, 40, 114, 113], phân chia tài nguyên [128], hệ thống tư vấn [98], tóm tắt dữ liệu [99], chọn bộ kích hoạt [107]... Một dạng tổng quát của các bài toán này là Phủ Submodular (SC) được định nghĩa như sau:

Định nghĩa 4.1 (Bài toán SC)**.** Cho một hàm submodular đơn điệu không âm $f : 2^V \mapsto \mathbb{R}_+$ và một ngưỡng $\alpha > 0$, bài toán yêu cầu tìm một tập lời giải $S \subseteq V$ với lực lượng nhỏ nhất sao cho $f(S) \geq \alpha$.

Trong chương này, luận án mở rộng nghiên cứu giải bài toán SC trên lưới nguyên, với hàm mục tiêu submodular được mở rộng thành hàm *DR-submodular*

(Disminishing Return Submodular). Bài toán SC được tổng quát hoá thành Phủ DR-Submodular (DRSC) được phát biểu như sau:

Định nghĩa 4.2 (Bài toán DRSC)**.** Cho một hàm DR-submodular đơn điệu $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, một số nguyên dương B là giá trị lớn nhất trên một trục tọa độ bất kỳ của một vec-tơ trong $\mathbb{Z}_+^V$, và một ngưỡng $\alpha > 0$. Bài toán cần tìm vec-tơ lời giải $\mathbf{x} \leq \mathbf{B}$ có lực lượng nhỏ nhất sao cho hàm mục tiêu không nhỏ hơn α, hay:

$$\min \ \|\mathbf{x}\|_1 \quad \text{s.t} \quad f(\mathbf{x}) \geq \alpha, \mathbf{0} \leq \mathbf{x} \leq \mathbf{B}, \tag{4.1}$$

trong đó $\mathbf{B} = B \cdot \mathbf{1}$ và $\|\mathbf{x}\|_1 = \sum_{e \in V} \mathbf{x}(e)$.

Đối với hàm submodular là hàm tập hợp, một phần tử e của tập cơ sở chỉ được chọn một lần. Để phần tử e được chọn lại nhiều lần cần có sự mở rộng tập hợp trên lưới nguyên. Khi đó, tập hợp trở thành đa tập hợp $\{\mathbf{x}\}$, tương ứng với một vec-tơ $\mathbf{x} \in \mathbb{Z}_+^V$. Số lần phần tử e xuất hiện gọi là $\mathbf{x}(e)$. Ví dụ, cho tập cơ sở $V = \{e_1, e_2, e_3\}$, vec-tơ $\mathbf{x} = (2, 4, 0)$ trên lưới nguyên biểu diễn đa tập hợp $\{\mathbf{x}\}$ tương ứng $\{e_{11}, e_{12}, e_{21}, e_{22}, e_{23}, e_{24}, e_{21}\}$. Với một tập con $A \subseteq V$ bất kỳ, kích cỡ của $\mathbf{x}$ là $\|\mathbf{x}\|_1 = \sum_{e \in A} \mathbf{x}(e)$, là tổng số phần tử có mặt trong đa tập hợp $\{\mathbf{x}\}$ tương ứng, trong ví dụ trên là 7 phần tử.

4.1.2. Hàm mục tiêu và một số quy ước quan trọng

Khi hàm submodular f là hàm tập hợp, $f : 2^V \mapsto \mathbb{R}_+$, nó thỏa mãn tính chất lợi nhuận hiệu suất giảm dần (Định nghĩa 1.4). Tuy nhiên, trên lưới nguyên, tính chất submodular và tính chất lợi nhuận hiệu suất giảm dần không còn là một. Submodular lúc này được phân tách thành lợi nhuận hiệu suất giảm dần và lattice submodular [128].

- Lợi nhuận hiệu suất giảm dần trên lưới nguyên

Lợi nhuận hiệu suất giảm dần (Diminishing Return submodular), thường gọi là *DR-submodular*, là tính chất tổng quát của submodular trên lưới nguyên [128]. Tính chất này mạnh hơn và phổ dụng hơn so với lattice submodular vì nó liên quan trực tiếp đến lợi ích thu về khi thêm một phần tử vào một tập trên lưới nguyên [128].

Để phát biểu tính chất DR-submodular, ta cần xây dựng các quy ước tập hợp như sau:

Cho một số nguyên dương $k \in \mathbb{N}$, ký hiệu $[k]$ đại diện cho tập $\{1, \ldots, k\}$. Cho tập cơ sở $V = \{e_1, \ldots, e_n\}$, ta ký hiệu $\mathbf{x}(e)$ là giá trị tọa độ của vec-tơ $\mathbf{x} \in \mathbb{Z}_+^V$ ứng với phần tử e nào đó. Ta cũng ký hiệu *vec-tơ đơn vị thứ* e (e-th unit vec-tơ) là χ_e với $\chi_e(t) = 1$ nếu $t = e$ và $\chi_e(t) = 0$ nếu $t \neq e$. Để nhất

quán, ký hiệu $f(\chi_e|\mathbf{x}) = f(\mathbf{x}+\chi_e) - f(\mathbf{x})$ là lợi nhuận biên (*marginal gain*) khi thêm một phần tử e vào trong tập hợp do vec-tơ $\mathbf{x}$ biểu diễn. Đồng thời, ký hiệu $f(\mathbf{x}|\mathbf{y}) = f(\mathbf{x}+\mathbf{y}) - f(\mathbf{y})$ là lợi ích đóng góp của vec-tơ $\mathbf{y}$ vào vec-tơ $\mathbf{x}$.

Thêm vào đó, với mọi vec-tơ $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$, nói rằng $\mathbf{x} \leq \mathbf{y}$ nếu và chỉ nếu $\mathbf{x}(e) \leq \mathbf{y}(e), \forall e \in V$. Lúc này, hàm $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$ thỏa mãn tính chất DR-submodular khi và chỉ khi:

$$f(\mathbf{x}+\chi_e) - f(\mathbf{x}) \geq f(\mathbf{y}+\chi_e) - f(\mathbf{y}), \tag{4.2}$$

với $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V, \mathbf{x} \leq \mathbf{y}$.

- Tính chất đơn điệu tăng xét trên lưới nguyên

Bài toán DRSC đang được xét với f là hàm đơn điệu. Hàm f là hàm đơn điệu tăng khi và chỉ khi:

$$f(\mathbf{x}) \leq f(\mathbf{y}) \; \forall \mathbf{x} \leq \mathbf{y}. \tag{4.3}$$

Để thuận tiện trong xây dựng các thuật toán và phân tích lý thuyết, Bảng 4.1 sẽ tóm tắt các ký hiệu thường dùng trong chương này. Ngoài ra, để giải bài toán này, ta cũng cần giả sử cách tính hàm f đã cho trước thông qua một hộp đen có thể ước lượng giá trị (ước lượng truy vấn của hàm mục tiêu). Tức là với mọi vec-tơ $\mathbf{x}$ bất kỳ, ta đều tính được $f(\mathbf{x})$. Không giảm tổng quát, giả sử $f(\mathbf{0}) = 0$.

4.2. Ứng dụng của bài toán

Bài toán có ứng dụng khi một phần tử cho lợi ích "tốt" được lựa chọn nhiều lần. Một số tình huống thực tiễn được đặt ra cần giải quyết với DR-submodular được xem xét dưới đây.

1. **Đặt cảm biến.** Giả sử có một số loại cảm biến với các mức năng lượng khác nhau. Giả định có sự đánh đổi đơn giản giữa thông tin thu được và chi phí tốn kém. Các cảm biến ở mức năng lượng cao có thể thu thập một lượng đáng kể thông tin, nhưng phải trả chi phí cao cho việc đặt chúng. Cảm biến mức năng lượng thấp có thể được đặt với chi phí thấp, nhưng chúng chỉ có thể thu thập thông tin hạn chế. Trong kịch bản này, chúng ta muốn quyết định loại cảm biến nào sẽ được đặt tại mỗi vị trí, thay vì chỉ quyết định có nên đặt một cảm biến tại vị trí đó hay không. Hay một vị trí có thể đặt nhiều cảm biến để thu được lượng thông tin tương ứng, nhưng chi phí rẻ hơn so với đặt một loại cảm biến khác tốn chi phí hơn. Một kịch bản như vậy nằm ngoài các mô hình dựa trên các hàm tập hợp submodular. Ta cần phải xét giải bài toán trên lưới nguyên.

2. **Tối ưu phân bổ tài nguyên, ngân sách.** Một tình huống tương tự cũng nảy

Ký hiệu	Mô tả
V	Tập cơ sở $V = \{e_1, e_2, .., e_n\}$
n	Kích cỡ của tập V
ϵ, λ	Các tham số chính xác cho trước, $\epsilon, \lambda \in (0, 1)$
$[m]$	Tập hợp gồm m số nguyên dương, $[m] = \{1, 2, \ldots, m\}$
$\mathbf{x}$	Vec-tơ bất kỳ trên $\mathbb{Z}_+^V$
$\mathbf{y}_i$	Vec-tơ được tạo bởi tiền tố của a_i trong dãy A, $\mathbf{y}_i = \sum_{j=1}^{i} \chi_{a_j}$
$f(\mathbf{x})$	Giá trị của hàm mục tiêu
$\mathbf{o}$, opt	$\mathbf{o}$ là vec-tơ lời giải tối ưu, opt $= f(\mathbf{o})$
χ_e	Vec-tơ đơn vị của phần tử $e \in V$
$\{\mathbf{x}\}$	Đa tập hợp (multiset) chứa tất cả các phần tử của vec-tơ $\mathbf{x}$, trong đó mỗi $e \in V$ xuất hiện nhiều lần
$\mathbf{x}(e)$	Giá trị toạ độ của phần tử e trong vec-tơ $\mathbf{x}$ bất kỳ, với $e \in V$
$\|\|\mathbf{x}\|\|_1$	Kích cỡ chuẩn của $\mathbf{x}$ với $\|\|\mathbf{x}\|\|_1 = \sum_{e \in V} \mathbf{x}(e)$
$\wedge$	Phép toán lấy giá trị nhỏ nhất trên từng tọa độ, tức là $\forall \mathbf{x}, \mathbf{y}$ và $\forall e \in V$, $\mathbf{x} \wedge \mathbf{y}(e) = \min\{\mathbf{x}(e), \mathbf{y}(e)\}$
$\vee$	Phép toán lấy giá trị lớn nhất trên từng tọa độ, tức là $\forall \mathbf{x}, \mathbf{y}$ và $\forall e \in V$, $\mathbf{x} \vee \mathbf{y}(e) = \max\{\mathbf{x}(e), \mathbf{y}(e)\}$
$\mathbf{x} + \mathbf{y}$	Tổng 2 vec-tơ với đa tập hợp $\{\mathbf{x} + \mathbf{y}\}$ trong đó các đỉnh $e \in V$ xuất hiện $\mathbf{x}(e) + \mathbf{y}(e)$ lần
α	Ngưỡng tối thiểu của hàm f
θ	Ngưỡng tham lam giảm dần của thuật toán
B	Lực lượng tối đa trên 1 trục tọa độ, $B > 0$
$\mathbf{0}$	Vec-tơ 0, trong đó mỗi phần tử $e \in V$ đều xuất hiện 0 lần, $\mathbf{0}(e) = 0$
$\mathbf{1}$	Vec-tơ biểu diễn tập V, trong đó mỗi phần tử của V đều được chọn
$\mathbf{B}$	Vec-tơ chặn trên của mọi vec-tơ $\mathbf{x}$ trên lưới nguyên, $\mathbf{B} = B.\mathbf{1}$
$f(\chi_e \vert \mathbf{x})$	$f(\chi_e \vert \mathbf{x}) = f(\mathbf{x} + \chi_e) - f(\mathbf{x})$: lợi nhuận biên (marginal gain) khi thêm một phần tử e vào $\mathbf{x}$
$f(\mathbf{x} \vert \mathbf{y})$	$f(\mathbf{x} \vert \mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{y})$
A	Dãy các phần tử được rút ngẫu nhiên liên tiếp, $A = [a_1, \ldots, a_m]$
X	Dãy các phần tử được lấy từ V
X_i	Dãy các phần tử được lấy từ X để bổ sung vào $\mathbf{x}$
R	Tập con được lấy ngẫu nhiên từ X
N	Số phần tử của R được lấy ra từ X
I	Tập hợp các ví trí σ_i sắp xếp tăng dần của các xâu con R_{σ_i} của R
C	Tập hợp giới hạn các phần tử v là ước lượng của opt

Bảng 4.1: Các ký hiệu thường dùng trong bài toán DRSC

sinh trong bài toán phân bổ ngân sách tối ưu. Trong vấn đề này, chúng ta muốn phân bổ ngân sách giữa các nguồn quảng cáo sao cho (ít nhất) một số lượng khách hàng nhất định bị ảnh hưởng trong khi giảm thiểu tổng ngân sách. Một lần nữa, chúng ta phải quyết định nên dành bao nhiêu ngân sách cho mỗi nguồn quảng cáo. Do đó, hàm tập hợp không thể giải quyết được tình huống này.

Các tình huống tương tự như trên thúc đẩy các nhà nghiên cứu tìm hiểu các phiên bản tổng quát hóa của tính submodular và lợi nhuận hiệu suất giảm dần được

định nghĩa trên các *lưới nguyên (Integer lattice)*, trở thành DR-submodular. Nếu ta xét thêm điều kiện giới hạn về lực lượng ít nhất mà vẫn thu được tối thiểu lượng thông tin α, bài toán trở thành thể hiện của bài toán DRSC.

4.3. Các thách thức của bài toán

Soma và Yoshida [128] lần đầu tiên mở rộng hàm f trên lưới nguyên, tức là 2^V_+ trở thành $\mathbb{Z}^V_+$, và xem xét submodular với các tính chất tổng quát hơn trên miền này. Các ông cũng khẳng định tính chất lợi nhuận hiệu suất giảm dần trên lưới nguyên, DR-submodular, là phiên bản tổng quát nhất của submodular trên lưới nguyên.

Việc nghiên cứu giải bài toán Phủ Submodular đơn điệu trên lưới nguyên là một bài toán mới hiện nay, và có nhiều thách thức đặt ra:

- Mặc dù đã có nhiều công trình nghiên cứu để giải SC, thách thức khi giải DRSC gặp phải là các tính chất của submodular sẽ không còn giữ nguyên khi xét trên lưới nguyên mà có sự mở rộng và khái quát hơn. Do vậy, các phương pháp thiết kế thuật toán theo hướng xấp xỉ lời giải tối ưu đã có đối với hàm tập hợp submodular chưa chắc đã áp dụng được trên lưới nguyên, hoặc không đạt được kết quả tốt như với hàm tập hợp;

- Bài toán SC là bài toán NP-khó, cho nên bài toán DRSC cũng là bài toán NP-khó;

- Vì không gian tìm kiếm là không gian nhiều chiều, $\mathbb{Z}^V_+$, nên số tổ hợp cần tìm sẽ tăng lên rất nhiều, ảnh hưởng tới thời gian tìm kiếm lời giải của thuật toán. Đồng thời, số lượng truy vấn lời gọi hàm cũng tăng nhanh theo;

- Đây là hướng nghiên cứu mới, cần đóng góp các thuật toán xấp xỉ có giá trị về mặt lý thuyết. Bài toán sẽ trở nên khó giải khi phải đảm bảo 2 ràng buộc, tìm lời giải tối ưu $\mathbf{o} \in \mathbb{Z}_+$ có kích thước (hoặc chi phí) nhỏ nhất và giá trị hàm mục tiêu của lời giải vượt ngưỡng cho trước.

Ta cũng có thể dẫn hàm f DR-submodular trên lưới nguyên về hàm tập hợp. Khái quát, giả sử có tập $V = \{e_1, e_2, e_3\}$, cho $r \in \mathbb{Z}_+$ là giới hạn số lần chọn nhiều nhất của từng phần tử của V. Một vec-tơ $\mathbf{x} = (2, 4, 0) \in \mathbb{Z}^V_+$ tức là có 2 phần tử e_1, 4 phần tử e_2 và không có phần tử e_3 nào được lựa chọn đưa vào vec-tơ. Khi đưa hàm f từ không gian lưới nguyên về hàm tập hợp, không gian tìm kiếm sẽ là tập cơ sở mới V' mới gồm B phần tử e_1, B phần tử e_2 và B phần tử e_3. Tuy nhiên, hệ quả là tập cơ sở sẽ có kích cỡ là $|V'| = B.|V|$ gồm $V' = \{e_{11}, e_{12}, \ldots, e_{1B}, e_{21}, e_{22}, \ldots, e_{2B}, e_{31}, e_{32}, \ldots, e_{3B}\}$. Như vậy không gian tìm kiếm sẽ tăng lên nhiều khi V là tập dữ liệu có kích thước lớn. Ngoài ra, cách dẫn về hàm tập hợp này cũng không thể làm việc được khi f là hàm lattice submodular [128].

Với bài toán DRSC, ta cũng có thể vận dụng trực tiếp các thuật toán đã giải cho SC sang DRSC nhờ vận dụng phương pháp dẫn về của Ene và Huy Nguyen [47], biểu diễn lưới nguyên thành đa tập hợp kích thước nB. Tuy nhiên, các thuật toán này chiếm $\Omega(nB)$ truy vấn để ước lượng hàm f. Điều này làm cho giải bài toán sẽ không còn là đa thức khi tập dữ liệu đầu vào có kích thước là $O(n \log B)$. Như vậy, phương pháp dẫn về chưa thể giải quyết DRSC một cách hiệu quả.

Từ các thách thức này, luận án đặt vấn đề nghiên cứu một hướng tiếp cận giải bài toán tối ưu hàm submodular phổ biến hiện nay đó là thiết kế thuật toán song song nhằm giảm thời gian tính toán. Đồng thời, luận án đề xuất *thuật toán xấp xỉ tiêu chí kép (bi-criteria approximation algorithm)* hiệu quả, cải tiến hơn so với các tác giả sử dụng giải pháp tương tự [63, 128] cho hai bài toán DRSC và SC.

Để xây dựng được thuật toán giải hiệu quả cần phải có các nghiên cứu về các bài toán SC và DRSC. Phần tiếp theo sẽ bàn về bối cảnh nghiên cứu hiện nay.

4.4. Các vấn đề nghiên cứu có liên quan

Để giải bài toán SC, các nhà nghiên cứu đã tập trung đề xuất các thuật toán xấp xỉ cạnh tranh dựa trên thủ tục tham lam [145, 63, 107, 98]. Tuy nhiên, bài toán là NP-khó và không thể tính xấp xỉ với tỉ lệ $(1 - o(1)) \log(n)$ trừ khi $NP \subset DTIME(n^{O(\log \log n)})$, trong đó n là kích thước của tập cơ sở [54]. Wolsey và cộng sự [145] lần đầu tiên đề xuất một thuật toán tham lam đơn giản nhưng hiệu quả với tỉ lệ xấp xỉ $1 + \ln(\max_{e \in V} f(e)/\beta)$ trong đó f là số thực và β là mức tăng lợi nhuận biên khác 0 nhỏ nhất của bất kỳ phần tử nào được thuật toán tham lam thêm vào lời giải.

Sau đó, Wan và cộng sự [139] đã giải bài toán Phủ Submodular với chi phí tối thiểu (Minimum Cost Submodular Cover - MCSC). Đó là một phiên bản tổng quát của bài toán SC trong đó mỗi phần tử e có chi phí dương $c(e)$ và hàm chi phí c là hàm cộng, nghĩa là $c(S) = \sum_{e \in S} c(e)$. Bài toán yêu cầu tìm nghiệm $S \subseteq V$ sao cho $c(S)$ là nhỏ nhất. Các tác giả đã đề xuất một thuật toán tham lam với tỉ lệ xấp xỉ là $\rho \ln(\max_{e \in V} f(e))$, trong đó $\rho = \min \sum_{e \in S, S \subseteq V} c(e)/c(S)$ được gọi là độ cong (curvature) của hàm c. Các tác giả trong [63] đã đề xuất một thuật toán tham lam khác trả về giải pháp xấp xỉ $(1 + \ln(\alpha/\epsilon), 1 - \epsilon)$ cho bài toán MCSC.

Gần đây, Soma và Yoshida đã giới thiệu bài toán SC trên lưới nguyên [128]. Bên cạnh việc nêu 2 tính chất lattice submodular và DR-submodular, các tác giả cũng đã đề xuất một thuật toán xấp xỉ với 2 tỉ lệ xấp xỉ được đưa ra là σ_1 cho hàm mục tiêu, và σ_2 cho hàm chi phí. Cho nên, các thuật toán được gọi là thuật toán xấp xỉ tiêu chí kép. Thuật toán của họ thực hiện trong $O(n \log(Bn) \log B)$ truy vấn.

Tuy nhiên, họ xem xét bài toán tổng quát của SC trên lưới nguyên với hàm mục tiêu f là DR-submodular đơn điệu tăng và hàm chi phí là hàm *giả cộng tính (subadditive)* $c : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, có nghĩa là, $c(\mathbf{x} + \mathbf{y}) \leq c(\mathbf{x}) + c(\mathbf{y})$ với mọi $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$. Bài toán họ nghiên cứu yêu cầu cần tối thiểu hóa chi phí $c(\cdot)$ để $f \geq \alpha$ với $\alpha > 0$ là một ngưỡng cho trước. Thuật toán của họ cho lời giải $\mathbf{x}$ thoả mãn $c(\mathbf{x}) \geq (1 + \epsilon)\rho(1 + \log(d/\beta)c(\mathbf{o}))$, và $f(\mathbf{x}) \geq (1 - \lambda)\alpha$, với ρ là một độ cong của c, $d = \max_{e \in V} f(\chi_e)$ là giá trị lớn nhất của f trên tất cả các vec-tơ đơn vị chuẩn và β là giá trị nhỏ nhất của số gia dương của f trong miền khả thi, $\epsilon, \lambda > 0$ là các tham số đầu vào. Thuật toán của họ cần thời gian chạy là $O(n \log(nBc_{max}/(\lambda c_{min})) \log(B)/\epsilon)$, với c_{min}, c_{max} lần lượt là các chi phí nhỏ nhất, lớn nhất của các đỉnh.

Do sự tăng trưởng theo cấp số nhân của dữ liệu trong các ứng dụng SC và DRSC (ví dụ: tăng trưởng của người dùng trong các mạng xã hội), cần phải đưa ra các thuật toán hiệu quả không chỉ cung cấp các đảm bảo xấp xỉ mà còn có thể mở rộng quy mô dữ liệu lớn về kích cỡ. Một phương pháp đơn giản để cải thiện hiệu quả của thuật toán là giảm độ phức tạp của truy vấn, qua đó góp phần giảm thời gian chạy vì độ phức tạp của truy vấn là vấn đề chính chi phối thời gian chạy của thuật toán.

Tuy nhiên, khi kích cỡ của dữ liệu tăng lên, phương pháp này có thể không đủ. *Thuật toán song song hóa* là một thiết kế tốt được lựa chọn trong nhiều công bố gần đây [11, 2, 1]. Thiết kế song song dựa trên khái niệm về *độ phức tạp song song (adaptivity hoặc adaptive comlexity)* được Balkanski phát biểu như sau:

Định nghĩa 4.3 (Độ phức tạp song song [10])**.** Cho một cách ước lượng hàm f, độ phức tạp song song của một thuật toán là số vòng lặp tối thiểu cần dùng để trong mỗi vòng lặp đó thuật toán tạo ra một số lượng đa thức các truy vấn tới ước lượng hàm f một cách độc lập với nhau.

Đây là phép đo quan trọng để xác định hiệu suất của thuật toán song song thiết kế cho bài toán tối ưu hàm submodular. Độ phức tạp song song là số vòng tuần tự tối thiểu trong đó mỗi vòng tạo ra một số truy vấn f được tiến hành song song, độc lập với nhau. Do đó, thuật toán có độ phức tạp song song thấp sẽ giảm được số vòng tuần tự cần phải thực hiện. Cách tiếp cận này cho hiệu quả để cải thiện các thuật toán, góp phần làm giảm thời gian chạy cho bài toán tối ưu hàm submodular.

Đối với bài toán SC, các thuật toán tham lam được thiết kế tuần tự thường có độ phức tạp song song là $\Omega(|S|)$ và có thể đạt tới $O(n)$, trong đó S là nghiệm trả về và $|S|$ có thể lớn tới n. Do đó, các thuật toán này mất quá nhiều thời gian để

tìm kiếm được lời giải. Ngoài ra, một thuật toán luồng để giải SC đã được đề xuất trong [108]. Trong công bố của họ, thuật toán đã trả về một giải pháp xấp xỉ tiêu chí kép $(1-1/\log(1/\epsilon), 2\log(1/\epsilon))$ nhưng nó vẫn yêu cầu số vòng để song song là $O(n)$. Thêm vào đó, Fahrbach và cộng sự [52] đã đề xuất thuật toán ngẫu nhiên có tỉ lệ xấp xỉ $O(\log \alpha)$ theo kỳ vọng nhưng yêu cầu $O(\log(n\log(\alpha))\log(\alpha))$ vòng lặp cho SC.

Gần đây, Ran và cộng sự [119] lần đầu tiên cung cấp một thuật toán song song hiệu quả và ngẫu nhiên cho bài toán SC. Trong đó f có giá trị nguyên, chạy trong $O\left(\log(n)\log(m)\log^2(mn)/\epsilon^4\right)$ vòng, cho tỉ lệ xấp xỉ $H(\min\{\sigma,\alpha\})/(1-5\epsilon)$ trong đó H là số Harmornic, $m = f(V) \geq \alpha$ và $\epsilon \in (0, 1/5)$. Trong trường hợp này, thuật toán của tác giả cải tiến đáng kể độ phức tạp song song lên tới $O(\log n)$ cho các lời giải xấp xỉ cùng theo tiêu chí kép. Tuy nhiên, họ chỉ giải với hàm f có giá trị nguyên.

Đối với DRSC, các công trình của Soma và Yoshida [128] là công trình đầu tiên nghiên cứu về DRSC bằng cách xem xét hàm DR-submodular trên lưới nguyên $\mathbb{Z}_+^V$ (DRSC) với c là hàm giả cộng tính. Vì DRSC tổng quát hóa SC, nên nó kế thừa độ khó của SC đã được đề cập trong [54]. Trong [128], các tác giả đã giới thiệu thuật toán tham lam giảm ngưỡng trả về giải pháp xấp xỉ tiêu chí kép $((1+3\epsilon)\rho(1+\log(d/\beta)), 1-\lambda)$, trong đó $\epsilon, \lambda \in (0,1)$ là các tham số đầu vào, $d = \max_{e\in V} f(\chi_e)$ và ρ là độ cong của c. Thuật toán đã thực hiện số truy vấn là $O(\log(nBc_{max}/(\lambda c_{min}))\log(B)\log(n)/\epsilon)$. Tuy nhiên, nó cũng chạy trong các vòng tuần tự là $O(n\log(nBc_{max}/(\lambda c_{min}))\log(B)/\epsilon)$ nên khó có thể song song hóa một cách hiệu quả. Tuy nhiên, công trình của các tác giả nói trên là cơ sở để các nghiên cứu khác khi nghiên cứu tối đa hóa các hàm DR-submodular với các ràng buộc khác nhau [128, 129].

Một hướng nghiên cứu khác là của hai tác giả Ene và Huy Nguyen [47]. Họ đã giới thiệu một phương pháp dẫn về từ DR-submodular về submodular. Công bố của họ cho phép giải quyết bài toán tối đa hàm DR-submodular với ràng buộc lực lượng và chi phí trong thời gian đa thức với kích cỡ đầu vào là $O(n\log B)$. Ý tưởng của họ đó là biểu diễn $\mathbf{x}(e)$ dưới dạng nhị phân cho mỗi phần tử $e \in V$ và tạo ra một tập cơ sở có kích cỡ $O(n\log B)$.

Ta có thể áp dụng thuật toán cho độ phức tạp song song thấp của [52, 119] cho DRSC bằng phương pháp dẫn về của Ene và Huy [47]. Tuy nhiên, số vòng lặp tuần tự và độ phức tạp truy vấn của thuật toán có được lớn hơn rất nhiều so với thuật toán được đề xuất bởi luận án này, (xem Bảng 4.2 để so sánh). Gần đây, Ene và Huy đã công bố một thuật toán song song cho cực đại hàm DR-submodular [49]. Song thuật toán này cần một ước lượng tiên đoán cho gradient của phần mở rộng

liên tục của hàm DR-submodular. Vì vậy, nó không thể áp dụng một cách trực tiếp cho bài toán DRSC trên lưới nguyên được.

Như vậy, có rất nhiều thuật toán tích cực cho SC và DRSC trong nghiên cứu; tuy nhiên, một số đó có độ phức tạp song song cao khi chúng liên tục phải tính toán các hàm mục tiêu [145, 63, 98, 107], trong khi một số khác chỉ áp dụng cho SC [52, 119]. Hiện nay, chưa có thuật toán xấp xỉ nào cho DRSC với độ phức tạp song song thấp. Điều này đã thúc đẩy tác giả và nhóm nghiên cứu đề xuất một thuật toán đảm bảo hiệu suất làm giảm đáng kể độ phức tạp truy vấn và độ phức tạp song song so với các thuật toán hiện đại đối với cả hai bài toán DRSC và SC.

4.5. Thuật toán xấp xỉ cho bài toán DRSC

4.5.1. Kết quả mới của luận án

Việc tìm được lời giải tối ưu sao cho chi phí nhỏ nhất với ràng buộc hàm $f(\cdot)$ là hàm submodular làm cho bài toán trở nên khó khăn hơn khi phải thỏa mãn cả hàm chi phí và hàm submodular. Do đó, luận án sử dụng thuật toán xấp xỉ tiêu chí kép để tìm một thuật toán xấp xỉ hiệu quả giải bài toán DRSC. Thuật toán xấp xỉ tiêu chí kép được đưa ra khi nới lỏng các ràng buộc của bài toán, cho lời giải có hai tỉ lệ xấp xỉ, được định nghĩa như sau:

Định nghĩa 4.4 (Thuật toán xấp xỉ tiêu chí kép (σ_1, σ_2))**.** Thuật toán là xấp xỉ tiêu chí kép với tỉ lệ (σ_1, σ_2) cho bài toán DRSC khi nó trả lại lời giải $\mathbf{x}$ thỏa mãn $\|\mathbf{x}\|_1 \leq \sigma_1 \cdot \|\mathbf{o}\|_1$ và $f(\mathbf{x}) \geq \sigma_2 \cdot \alpha$ với $\sigma_1 > 1, \sigma_2 > 0$, và $\mathbf{o}$ là lời giải tối ưu.

Để dễ tiếp cận với phương pháp giải, trước tiên luận án giới thiệu một phiên bản đơn giản hóa của thuật toán được đề xuất với giả định opt đã biết (Thuật toán 11), với opt là kích thước của giải pháp tối ưu. Sau đó, phần giả định này sẽ được loại bỏ trong phiên bản chính (Thuật toán 12) khi xấp xỉ opt trong một khoảng thích hợp.

Các đóng góp của luận án có thể kể đến như sau: Luận án đề xuất một thuật toán xấp xỉ tiêu chí kép $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$ cho bài toán DRSC, với $\epsilon, \lambda \in (0, 1)$ là các đầu vào đã cho. Thuật toán cần $O\left((n + \log(n)\log(B))\log(nB)\right)$ truy vấn và chạy với độ phức tạp song song là $O(\log n)$. Như vậy, thuật toán được đề xuất không chỉ cải tiến được tỉ lệ xấp xỉ mà còn thực sự giảm đáng kể độ phức tạp truy vấn và độ phức tạp song song so với các thuật toán tiên tiến hiện nay [128, 52, 119, 47]. Các đóng góp được mô tả chi tiết như dưới đây:

1. **Về mặt chất lượng lời giải.** Thuật toán cho lời giải xấp xỉ với tỉ lệ hằng số là $(1 + \epsilon)(1 + \log(1/\lambda))$ và giá trị hàm f gần tùy ý với α (trong giới hạn sai số $\lambda > 0$). Như vậy, thuật toán của tác giả cho tỉ lệ xấp xỉ tốt hơn so với thuật

toán tất định tốt nhất được đề xuất bởi Soma và Yoshida [128]. Thuật toán của họ thực tế phụ thuộc vào $O(\log d)$, với $d = \max_{e \in V} f(\chi_e)$). Thuật toán được nêu trong luận án cũng đảm bảo xấp xỉ tốt hơn thuật toán ngẫu nhiên tốt nhất hiện nay nếu kết hợp hai phương pháp được đề xuất bởi Ran [119] về thuật toán song song và phương pháp của Ene và Huy Nguyen [47] dẫn DRSC về SC. Tỉ lệ xấp xỉ của thuật toán có được khi kết hợp các phương pháp của họ sẽ phụ thuộc vào $O(H(\min\{\max_{e \in S} f(e), \alpha\}))$. Ngoài ra, lưu ý rằng thuật toán của họ chỉ giải quyết trường hợp hàm mục tiêu mang giá trị nguyên.

2. **Về độ phức tạp.** Thuật toán ngẫu nhiên cho độ phức tạp song song tốt nhất là thuật toán kết hợp giữa [119] và [47]. So với các kết hợp này, thuật toán trong luận án cho độ phức tạp song song thấp hơn 1 hệ số là $\Omega(\log(m) \log^2(mn \log B)$ $(1 + \log(\log B)/\log(n)))$. Đồng thời, thuật toán cũng cho độ phức tạp truy vấn thấp hơn 1 hệ số là $\Omega(\min\{\log(n) \log(B)/\log(nB), n\} \cdot \log(m) \log^2(mn \log(B)))$ với $m = f(B \cdot \mathbf{1})$. So sánh với thuật toán tất định cho độ phức tạp truy vấn tốt nhất của Soma và Yoshida [128], thuật toán cho độ phức tạp song song và độ phức tạp truy vấn ít hơn 1 hệ số lần lượt là $\Omega(\min\{n/\log(n), \log(B)\})$ và $\Omega(n \log(B)(1 + \log(B)))$.

3. **Về giải bài toán SC.** Để giải bài toán SC, thuật toán cho lời giải xấp xỉ tiêu chí kép $((1+\epsilon)(1+\log 1/\lambda), 1-\lambda)$ với độ phức tạp song song là $O(\log n)$ và độ phức tạp truy vấn là $O(n \log n)$. Đây là các đảm bảo lý thuyết có giá trị và vượt trội hơn so với các thuật toán xấp xỉ hiện nay được đề xuất bởi [119, 52].

4. **Kỹ thuật được sử dụng.** Để giảm độ phức tạp song song, luận án đã vận dụng ý tưởng kỹ thuật *Lấy mẫu song song tuần tự (adaptive sequential sampling)* của Balkanski nêu trong [10] nhưng cần một số biến đổi phù hợp: (1) lấy mẫu một dãy các phần tử trên lưới nguyên, và (2) lặp đi lặp lại thêm một vec-tơ vào vec-tơ lời giải khi nó có tỉ lệ lợi nhuận biên với kích cỡ của nó tốt.

Mặc dù các thuật toán ngẫu nhiên gần đây cho tỉ lệ song song thấp của Ran và cộng sự [119], và của Ene và Huy Nguyen [47] cũng vận dụng ý tưởng lấy mẫu song song tuần tự, nhưng việc sinh ra các mẫu trên tập cơ sở V của các thuật toán này làm cho độ phức tạp của nó tăng lên khi áp dụng trên lưới nguyên. Bên cạnh đó, thuật toán của họ tái sử dụng phương pháp của Blelloch [14] chỉ áp dụng được cho bài toán Tập phủ cực đại (Set Cover), và nó cũng chỉ làm việc được với hàm mục tiêu có giá trị nguyên.

Khác với các công bố trên đây, luận án sử dụng một thuật toán lấy mẫu để sinh ra một dãy các phần tử thỏa mãn ràng buộc trên lưới nguyên. Sau đó, các *tiền tố (prefix)* của dãy phần tử này sẽ được lấy ra và đánh giá, so sánh dựa trên *tỉ lệ giữa lợi nhuận biên của từng tiền tố* khi thêm vào tập lời giải *với kích cỡ* của nó.

Bên cạnh đó, luận án kết hợp với *phương pháp ngưỡng giảm dần (decreasing threshold)*, là phương pháp được đề xuất bởi Bandanidiyuru và cộng sự [6]. Phương pháp áp dụng trong bài để tạo ra một giới hạn các phần tử để chất lượng lời giải tốt hơn và phù hợp khi hàm mục tiêu là số thực. Theo cách này, thuật toán được đề xuất chỉ cần số vòng lặp tuần tự là $O(\log(n))$. Tại mỗi vòng, nó đồng thời chọn tiền tố tốt nhất để thêm vào giải pháp của mình và loại bỏ các phần tử có tỉ lệ thấp cho các vòng sau.

Để giảm thêm độ phức tạp của truy vấn, luận án sử dụng phương pháp lấy mẫu để lấy một tập hợp con ngẫu nhiên đồng nhất của X (tập hợp các phần tử còn lại) thay vì tất cả các phần tử trong X. Cuối cùng, bằng cách tìm kiếm trên một phạm vi thích hợp và đặt điều kiện dừng hợp lý, thuật toán đảm bảo sự cân bằng giữa đảm bảo hiệu suất của giải pháp và độ phức tạp thích ứng.

Bảng 4.2 cung cấp cái nhìn tổng quan khi so sánh các thuật toán cho bài DRSC và SC. Bảng này so sánh các thuật toán xấp xỉ giữa $\mathbf{BA}$ là thuật toán được đề xuất của luận án với một số thuật toán tiên tiến. Các thuật toán được so sánh theo các tiêu chí: tỉ lệ xấp xỉ, độ phức tạp truy vấn, độ phức tạp song song. Ngoài ra, các thuật toán còn được phân loại theo thuật toán ngẫu nhiên hay thuật toán tất định. Ký hiệu *TT* là Thuật toán. Các bộ có dạng (x, y) nhằm ký hiệu cho thuật toán tiêu chí kép cho tỉ lệ (x, y). Có $n' = n\log(B)$, $H(\cdot)$ là số Harmonic, $m = f(V)$ cho SC, và $m = f(B \cdot \mathbf{1})$ cho DRSC, β là lợi nhuận biên nhỏ nhất của bất kỳ phần tử nào được chọn bởi thuật toán, $d = \max_{e \in V} f(\chi_e)$. ϵ, λ là các tham số đầu vào. Các ký hiệu [52] + [47] và [119] + [47] ngụ ý sử dụng kết hợp các thuật toán của [52, 119] với phương pháp dẫn về của [47]. Lưu ý rằng các thuật toán trong [52, 119] chỉ hữu dụng khi f là số nguyên.

Để phân tích các đảm bảo lý thuyết khi thiết kế thuật toán, luận án căn cứ vào Bổ đề về giới hạn Chernoff như sau:

Bổ đề 4.1 (Giới hạn Chernoff (Chernoff's bounds) [100])**.** *Đặt $X = \sum_{i=1}^{c} X_i$ là tổng của c biến ngẫu nhiên được lấy mẫu trong phân phối $[0, 1]$ với trung bình μ. Với bất kỳ $a > 0$, ta có:*

$$\Pr[X - c\mu \geq ac\mu] \leq e^{-\frac{a^2 c\mu}{2+a}} \tag{4.4}$$

$$\Pr[X - c\mu \leq -ac\mu] \leq e^{-\frac{a^2 c\mu}{2}}. \tag{4.5}$$

4.5.2. Thuật toán với opt đã biết: $\mathbf{AdaptDRSC}$

Đầu tiên, một phiên bản giả sử đã biết giá trị tối ưu nhằm đơn giản các bước tính toán trung gian được xây dựng, thuật toán $\mathbf{AdaptDRSC}$. Thuật toán nhận

TT	Tỉ lệ xấp xỉ	Độ phức tạp truy vấn	Độ phức tạp song song	Loại
Các thuật toán cho DRSC				
[128]	$((1+3\epsilon)(1+\log(\frac{d}{\beta})), 1-\lambda)$	$O\left(n' \log(nB)\right)$	$O\left(n' \log(nB)\right)$	Tất định
[52] + [47]	$O(\log(\alpha))$	$O(n' \log(n' \log(\alpha)) \log(\alpha))$	$O(\log(n' \log(\alpha)) \log(\alpha))$	Ngẫu nhiên
[119] + [47]	$\frac{H(\min\{\max_{e\in S} f(e), \alpha\})}{1-\epsilon}$	$O(n' \log(n') \log(m) \log^2(mn'))$	$O(\log(n') \log(m) \log^2(mn'))$	Ngẫu nhiên
BA	$((1+\epsilon)(1+\log(\frac{1}{\lambda})), 1-\lambda)$	$O((n+\log(n) \log(B)) \log(nB))$	$O\left(\log n\right)$	Ngẫu nhiên
Các thuật toán cho SC				
[145]	$1+\ln(\max_{e\in S} \frac{f(e)}{\beta})$	$\Omega(n\|S\|)$	$\Omega(\|S\|)$	Tất định
[63]	$\left(1+\log(\frac{\alpha}{\epsilon}), 1-\epsilon\right)$	$\Omega(n\|S\|)$	$\Omega(\|S\|)$	Tất định
[52]	$O(\log(\alpha))$	$O(n \log(n \log(\alpha)) \log(\alpha))$	$O(\log(n \log(\alpha)) \log(\alpha))$	Ngẫu nhiên
[119]	$\frac{H(\min\{\max_{e\in S} f(e), \alpha\})}{1-\epsilon}$	$O(n \log(n) \log(m) \log^2(mn))$	$O(\log(n) \log(m) \log^2(mn))$	Ngẫu nhiên
BA	$((1+\epsilon)(1+\log(\frac{1}{\lambda})), 1-\lambda)$	$O\left(n \log n\right)$	$O\left(\log n\right)$	Ngẫu nhiên

Bảng 4.2: So sánh các thuật toán xấp xỉ cho DRSC và SC. **BA** là thuật toán của luận án.

hai tham số đầu vào cho trước $\epsilon \in (0,1), \lambda \in (0,1)$ và một giá trị tiên đoán của giá trị tối ưu, v, sao cho $(1-5\epsilon_1)v \leq \mathsf{opt} \leq v$, với $\epsilon_1 = \epsilon/50$. Đặt $g(\mathbf{x}) = \min\{\alpha, f(\mathbf{x})\}$, dễ dàng thấy rằng g cũng là một hàm DR-submodular đơn điệu.

Thuật toán bao gồm hai vòng lặp và trong mỗi bước lặp của vòng lặp bên trong, thuật toán tạo ra một mẫu chuỗi ngẫu nhiên (random sequence sample) bằng cách gọi **RanSample** (Thuật toán 10) làm chương trình con. Chương trình con này nhận đầu vào là lời giải một phần $\mathbf{x}$, một tập hợp các phần tử khả thi X, số nguyên dương B và trả về một chuỗi $A = [a_1, \ldots, a_m]$. Mỗi phần tử a_i của A được rút ngẫu nhiên liên tiếp và đều trong số các phần tử a còn lại của X sao cho đảm bảo điều kiện $\mathbf{x} + \sum_{j=1}^{i-1} \chi_{a_j} + \chi_a \leq \mathbf{B}$.

Đặt $\mathbf{y}_i = \sum_{j=1}^{i} \chi_{a_j}$ là vec-tơ được tạo bởi phần trước của A từ vị trí i. Ta cần chọn một tiền tố của A với tỉ lệ *mức tăng lợi nhuận biên so với kích thước* cao để thêm vào vào lời giải hiện tại bằng cách tìm vị trí nhỏ nhất i^* sao cho $|X_{i^*}| < (1-\epsilon_1)|X|$, trong đó:

$$X_i \leftarrow \{e \in X : g(\chi_e | \mathbf{x} + \mathbf{y}_{i-1}) \geq \theta \text{ and } \mathbf{x} + \mathbf{y}_{i-1} + \chi_e \leq \mathbf{B}\}. \tag{4.6}$$

Thuật toán có với mọi $i \in [m]$ và loại bỏ ϵ_1 phần các phần tử trong X tại mỗi bước lặp. θ là ngưỡng để lọc ra được các phần tử có lợi nhuận biên cao. Tuy nhiên, thao

tác này có thể yêu cầu $n \cdot m = O(n^2 B)$ đánh giá hàm mục tiêu, tức là không còn đa thức phụ thuộc vào kích thước đầu vào. Nếu sử dụng thuật toán tìm kiếm nhị phân để tìm i^*, thì thuật toán này cần $O(n \log(nB))$ số lời gọi hàm trong các vòng tuần tự là $O(\log(nB))$. Để giảm được cả độ phức tạp song song và truy vấn, ta sẽ chọn ngẫu nhiên một tập con R có kích cỡ $N = O(\log(1/\epsilon)/\epsilon^2)$ từ X (dòng 6 của Thuật toán 11) và tìm số nguyên nhỏ nhất $\sigma_{i^*} \in I$ để $|R_{\sigma_{i^*}}| < (1 - 2\epsilon_1)|R|$ trên tập I thay vì $[m]$, có nghĩa là chọn ra:

$$R_i \leftarrow \{e \in R : g(\chi_e | \mathbf{x} + \mathbf{y}_{i-1}) \geq \theta \text{ and } \mathbf{x} + \mathbf{y}_{i-1} + \chi_e \leq \mathbf{B}\} \tag{4.7}$$

và giới hạn tập I:

$$I = \{\lceil (1 - \epsilon_1)^j nB \rceil : 1 \leq \lceil (1 - \epsilon_1)^j nB \rceil \leq nB, j \in \mathbb{N}\}. \tag{4.8}$$

Như vậy, lúc này thao tác cần $O(\log(nB) \log(1/\epsilon)/\epsilon^3)$ lời gọi hàm và 1 vòng lặp tuần tự. Thuật toán sẽ cập nhật $\mathbf{x}$ và tập khả thi X (dòng 11 trong Thuật toán 11).

Vị trí σ_{i^*} dẫn tới hai thuộc tính quan trọng của vòng lặp bên trong:

- Có ít nhất ϵ_0-phần của các phần tử bị loại bỏ khỏi X theo kỳ vọng (theo Bổ đề 4.2).

- vec-tơ $\mathbf{y}_{\sigma_{i^*}-1}$ có khả năng đóng góp giá trị cao cho giải pháp hiện tại (theo Bổ đề 4.4).

Vòng lặp bên trong kết thúc sau $O(\log(n)/\epsilon)$ bước lặp hoặc $f(\mathbf{x}) \geq (1 - \lambda)\alpha$ và chuyển sang bước lặp tiếp theo của vòng lặp bên ngoài. Nó tiếp tục cập nhật ngưỡng θ và kết thúc sau $O(\log(1/\lambda)/\epsilon)$ bước lặp của vòng lặp ngoài hoặc điều kiện $f(\mathbf{x}) \geq (1 - \lambda)\alpha$ được đáp ứng.

Algorithm 10 : $\mathbf{RanSample}(\mathbf{x}, B, X)$

Input: $\mathbf{x}$, a positive integer B, set X
Output: sample A

1: $A \leftarrow \emptyset, i \leftarrow 1$
2: **while** $X \neq \emptyset$ **do**
3: $\quad X \leftarrow \{a \in X : \mathbf{x} + \sum_{j=1}^{i-1} \chi_{a_j} + \chi_a \leq \mathbf{B}\}$
4: $\quad$ Draw a_i uniformly at random from X
5: $\quad A \leftarrow [a_1, \ldots, a_{i-1}, a_i]$
6: $\quad i \leftarrow i + 1$
7: **end while**
8: **return** A

Thuật toán $\mathbf{RanSample}$ trả về A là một dãy các phần tử khả thi được chọn ngẫu nhiên. Rõ ràng, thuật toán này không cần một lời gọi hàm f nào, nên độ phức tạp song song là 0. Thuật toán $\mathbf{AdaptDRSC}$ sẽ gọi đến $\mathbf{RanSample}$ và đưa ra một thiết kế song song khả thi.

Algorithm 11 : $\mathbf{AdaptDRSC}(f, B, \alpha, \epsilon_1, \lambda, v)$

Input: $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, a positive integer B, α, a guess of optimal value v, parameters $\epsilon_1 \in (0,1), \lambda \in (0,1)$

Output: vector $\mathbf{x}$

1: $t \leftarrow 1, \epsilon_1 \leftarrow \frac{\epsilon}{50}, \epsilon_0 \leftarrow \epsilon_1(1-\epsilon_1)$
2: **while** $f(\mathbf{x}) < (1-\lambda)\alpha$ and $t < \frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})$ **do**
3: $X \leftarrow V, \theta \leftarrow (1-5\epsilon_1)(\alpha - g(\mathbf{x}))/v, l \leftarrow 1$
4: **while** $f(\mathbf{x}) < (1-\lambda)\alpha$ and $l < \frac{1}{\epsilon_0}\log n$ **do**
5: $A = [a_1, a_2, \ldots, a_m] \leftarrow \mathbf{RanSample}(\mathbf{x}, B, X)$ // Generate a sequence sample by Algorithm 10
6: $N \leftarrow \frac{2+\epsilon_1}{\epsilon_1^2(1-\epsilon_1)}\log(\frac{1}{\epsilon_1})$, $R \leftarrow Sample(X, N)$ // Generate a random set from X
7: $I \leftarrow \{\sigma_1, \sigma_2, \ldots, \sigma_k : \sigma_1 < \sigma_2 < \ldots < \sigma_k\} \leftarrow \{\lceil(1-\epsilon_1)^j nB\rceil : 1 \leq \lceil(1-\epsilon_1)^j nB\rceil \leq nB, j \in \mathbb{N}\}$
8: $\mathbf{y}_l \leftarrow \sum_{j=1}^{l} \chi_{a_j}, l \in [m], \mathbf{y}_0 \leftarrow \mathbf{0}$
9: $R_{\sigma_i} \leftarrow \{e \in R : g(\chi_e|\mathbf{x} + \mathbf{y}_{\sigma_i - 1}) \geq \theta$ and $\mathbf{x} + \mathbf{y}_{\sigma_i - 1} + \chi_e \leq \mathbf{B}\}, \sigma_i \in I$
10: Find $\sigma_{i^*} \leftarrow \min\{\sigma_i \in I : |R_{\sigma_i}| < (1-2\epsilon_1)|R|\}$
11: Update the solution and feasible set $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{y}_{\sigma_{i^*}-1}, X \leftarrow X_{\sigma_{i^*}}, l \leftarrow l+1$
12: **end while**
13: $t \leftarrow t+1$
14: **end while**
15: **return** $\mathbf{x}$

Để phân tích các đảm bảo lý thuyết của thuật toán, ta có các bổ đề và định lý dưới đây.

Bổ đề 4.2. *Tại bất kỳ bước lặp nào của vòng lặp bên trong ta có:*

a) $X_{i+1} \subseteq X_i, i = 1, \ldots, m-1$, *với* $m = |A|$ *tại dòng 5 của thuật toán* $\mathbf{AdaptDRSC}$.

b) $\mathbb{E}[|X_{\sigma_{i^*}}|] \leq (1-\epsilon_0)|X|$ *và* $\mathbb{E}[|X_l|] \geq (1-\epsilon_1)(1-3\epsilon_1)|X|, \forall l \in [m], l \leq \sigma_{i^*-1}$.

Chứng minh. Chứng minh từng trường hợp:
- Chứng minh a): Với $e \in X_{i+1}$ bất kỳ, nó phải thỏa mãn điều kiện $g(\chi_e|\mathbf{x} + \mathbf{y}_i) \geq$

θ and $\mathbf{x} + \mathbf{y}_i + \chi_e \leq \mathbf{B}$. Do tính DR-submodular của f, ta có $g(\chi_e|\mathbf{x} + \mathbf{y}_{i-1}) \geq g(\chi_e|\mathbf{x} + \mathbf{y}_i) \geq \theta$ và $\mathbf{x} + \mathbf{y}_{i-1} + \chi_e \leq \mathbf{B}$, suy ra $e \in X_i$, nên $X_{i+1} \subseteq X_i$.
- Chứng minh b): Đặt $\mu_l = |X_l|/|X|$ và $\hat{\mu}_l = |R_l|/|R|$ với $l \in [m]$. Vì $\hat{\mu}_{\sigma_{i^*}} = |R_{\sigma_{i^*}}|/|R| < 1 - 2\epsilon_1$, áp dụng bất đẳng thức (4.5) trong Bổ đề 4.1 với $a = \epsilon_1$ và $c = N$, ta có:

$$\begin{aligned}
\Pr[|X_{\sigma_{i^*}}| \geq (1-\epsilon_1)|X|] &= \Pr[\mu_{\sigma_{i^*}} \geq 1 - \epsilon_1] \\
&\leq \Pr[\mu_{\sigma_{i^*}} \geq \frac{1-2\epsilon_1}{1-\epsilon_1}] \quad (\text{ Vì } 1 - \epsilon_1 \geq \tfrac{1-2\epsilon_1}{1-\epsilon_1}\ ) \\
&\leq \Pr[\mu_{\sigma_{i^*}} \geq \frac{\hat{\mu}_{\sigma_{i^*}}}{1-\epsilon_1}] = \Pr[\hat{\mu}_{\sigma_{i^*}} - \mu_{\sigma_{i^*}} \leq -\epsilon_1 \mu_{\sigma_{i^*}}] \\
&= \Pr[\hat{\mu}_{\sigma_{i^*}} N - N\mu_{\sigma_{i^*}} \leq -\epsilon_1 N \mu_{\sigma_{i^*}}] \\
&\leq e^{-\frac{\epsilon_1^2 \cdot N \cdot \mu_{\sigma_{i^*}}}{2}} \quad (\text{Bằng cách áp dụng bất đẳng thức (4.5) }) \\
&\leq e^{-\frac{\epsilon_1^2(1-\epsilon_1)N}{2}} = e^{-\frac{\epsilon_1^2(1-\epsilon_1)\frac{2+\epsilon_1}{\epsilon_1^2(1-\epsilon_1)}\log(\frac{1}{\epsilon_1})}{2}} \\
&< e^{-\log(\frac{1}{\epsilon_1})} \leq \epsilon_1.
\end{aligned}$$

Vì vậy $\Pr[|X_{\sigma_{i^*}}| \leq (1-\epsilon_1)|X|] \geq 1 - \epsilon_1$. Kết hợp với $|X_{\sigma_{i^*}}| \leq |X|$, ta có:

$$\begin{aligned}
\mathbb{E}[|X_{\sigma_{i^*}}|] &\leq (1-\epsilon_1)|X| \Pr\left(|X_{\sigma_{i^*}}| \leq (1-\epsilon_1)|X|\right) + |X_{\sigma_{i^*}}| \Pr(|X_{\sigma_{i^*}}| > (1-\epsilon_1)|X|) \\
&\leq (1-\epsilon_1)(1-\epsilon_1)|X| + \epsilon_1|X| \\
&= (1 - \epsilon_1(1-\epsilon_1))|X|.
\end{aligned}$$

Dễ thấy $R_{l+1} \subseteq R_l, \forall l \in [m]$. Nên $\hat{\mu}_l \geq 1 - 2\epsilon_1, l \leq \sigma_{i^*-1}$. Áp dụng bất đẳng thức (4.4) trong Bổ đề 4.1 với $a = \epsilon_1/\mu_l$ ta có:

$$\begin{aligned}
\Pr[|X_l| \leq (1-3\epsilon_1)|X|] &= \Pr[\mu_l \leq 1 - 3\epsilon_1] \\
&\leq \Pr[\mu_l \leq \hat{\mu}_l - \epsilon_1] \leq e^{-\frac{a^2 N \mu_l}{2+a}} \\
&\leq e^{-\frac{\epsilon_1^2 N}{2\mu_l + \epsilon_1}} \leq e^{-\frac{\epsilon_1^2 \frac{2+\epsilon_1}{\epsilon_1^2(1-\epsilon_1)}\log(\frac{1}{\epsilon_1})}{2\mu_l + \epsilon_1}} < e^{-\log(\frac{1}{\epsilon_1})} < \epsilon_1.
\end{aligned}$$

Vì vậy $\Pr[|X_l| \geq (1-3\epsilon_1)|X|] \geq 1 - \epsilon_1$, nên $\mathbb{E}[|X_l|] \geq (1-\epsilon_1)(1-3\epsilon_1)|X|$. □

Bổ đề 4.3. *Tại mỗi bước lặp của vòng lặp trong, đặt $l \leq \sigma_{i^*-1}, \forall l \in [m]$, ta có:*

$$\mathbb{E}[g(\chi_{a_l}|\mathbf{x} + \mathbf{y}_{l-1})] \geq (1-\epsilon_1)(1-3\epsilon_1)\theta.$$

Chứng minh. Đặt $X(l) = \{e \in X : \mathbf{x} + \mathbf{y}_{l-1} + \chi_e \leq \mathbf{B}\}$, với bất kỳ $l \leq \sigma_{i^*-1}$, từ

Bổ đề 4.2, ta có:

$$\mathbb{E}[|X_l|] \geq \mathbb{E}[|X_{\sigma_{i^*-1}}|] \geq (1-\epsilon_1)(1-3\epsilon_1)|X| \geq (1-\epsilon_1)(1-3\epsilon_1)|X(l)|.$$

Gọi một quy trình ngẫu nhiên là $\mathcal{H}_l$ để bắt sự kiện tạo tiền tố $A_l = [a_1, \ldots, a_l]$, a_l được rút ngẫu nhiên đồng nhất từ X_l. Cho trước $\mathcal{H}_{l-1}$, các tập hợp A_{l-1}, $X(l)$ và vec-tơ $\mathbf{x} + \mathbf{y}_{l-1}$, là các giá trị xác định, trong khi phần còn lại của chuỗi A là ngẫu nhiên.

Phần tử $e \in X(l)$ bất kỳ được lựa chọn đồng nhất với xác suất $p_e = 1/|X(l)|$ bằng cách sử dụng $\mathbf{RanSample}(\mathbf{x}, B, X)$, ta có:

$$\begin{aligned}
\mathbb{E}[g(\chi_{a_l}|\mathbf{x}+\mathbf{y}_{l-1})|] &= \mathbb{E}[g(\chi_{a_l}|\mathbf{x}+\mathbf{y}_{l-1})|\mathcal{H}_{l-1}] && (4.9)\\
&= \mathbb{E}[\mathbb{E}[g(\chi_{a_l}|\mathbf{x}+\mathbf{y}_{l-1})]|\mathcal{H}_{l-1}] && (4.10)\\
&= \mathbb{E}[\sum_{e\in X(l)} p_e g(\chi_e|\mathbf{x}+\mathbf{y}_{l-1})|\mathcal{H}_{l-1}] && (4.11)\\
&\geq \mathbb{E}[\sum_{e\in X_l} p_e g(\chi_e|\mathbf{x}+\mathbf{y}_{l-1})|\mathcal{H}_{l-1}] && (4.12)\\
&\geq \mathbb{E}[\sum_{e\in X_l} p_e \theta|\mathcal{H}_{l-1}] && (4.13)\\
&= \mathbb{E}[(1-\epsilon_1)(1-3\epsilon_1)\theta|\mathcal{H}_{l-1}] && (4.14)\\
&= (1-\epsilon_1)(1-3\epsilon_1)\theta. && (4.15)
\end{aligned}$$

Trong đó các bất đẳng thức (4.9) và (4.15) đảm bảo bởi luật tổng kỳ vọng. Bất đẳng thức (4.13) có được bởi định nghĩa của X_l. □

Bằng cách sử dụng Bổ đề 4.3, các tính chất của σ_{i^*} và tập I, thì vec-tơ $\mathbf{y}_{\sigma_{i^*}-1}$ có giá trị tỉ lệ cao đối với $\|\mathbf{y}_{\sigma_{i^*}-1}\|_1$ trong Bổ đề 4.4.

Bổ đề 4.4. *Tại bất kỳ bước lặp của vòng lặp trong, ta có:*

$$\mathbb{E}[g(\mathbf{y}_{\sigma_{i^*}-1}|\mathbf{x})] \geq (1-5\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_{\sigma_{i^*}-1}\|_1]. \quad (4.16)$$

Chứng minh. Từ định nghĩa của I, $\sigma_{i^*} - 1 \geq \sigma_{i^*-1} + 1$, giả sử $\sigma_{i^*} = \lceil a \rceil$, $a =$

$(1-\epsilon_1)^j nB$ với một vài $j \in \mathbb{N}$, ta có: $\sigma_{i^*-1} = \lceil (1-\epsilon_1)a \rceil$, vì vậy:

$$\begin{aligned}\|\mathbf{y}_{\sigma_{i^*}-1}\|_1 - \|\mathbf{y}_{\sigma_{i^*-1}}\|_1 &= \lceil a \rceil - (1 + \lceil (1-\epsilon_1)a \rceil) \\ &\leq \lceil a \rceil - (\lceil a \rceil + \lceil (-\epsilon_1 a) \rceil)) = -\lceil (-\epsilon_1 a) \rceil \\ &= \lfloor \epsilon_1 a \rfloor \leq \epsilon_1 a \leq \frac{\epsilon_1(\sigma_{i^*}-1)}{1-\epsilon_1} = \frac{\epsilon_1}{1-\epsilon_1}\|\mathbf{y}_{\sigma_{i^*}-1}\|_1.\end{aligned}$$

Vậy nên $\|\mathbf{y}_{\sigma_{i^*-1}}\|_1 \geq \frac{1-2\epsilon_1}{1-\epsilon_1}\|\mathbf{y}_{\sigma_{i^*}-1}\|_1$. Mặt khác, từ Bổ đề 4.3, ta có:

$$\begin{aligned}\mathbb{E}[g(\mathbf{y}_{\sigma_{i^*-1}}|\mathbf{x})] &= \mathbb{E}[\sum_{j=1}^{\sigma_{i^*-1}} g(\chi_{a_j}|\mathbf{x} + \mathbf{y}_{j-1})] \\ &\geq \mathbb{E}[(1-\epsilon_1)(1-3\epsilon_1)\theta \sum_{j=1}^{\sigma_{i^*-1}}] \\ &= (1-\epsilon_1)(1-3\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_{\sigma_{i^*-1}}\|_1].\end{aligned}$$

Áp dụng bất đẳng thức trên và kết hợp với tính đơn điệu của g, có:

$$\begin{aligned}\mathbb{E}[g(\mathbf{y}_{\sigma_{i^*}-1}|\mathbf{x})] &\geq \mathbb{E}[g(\mathbf{y}_{\sigma_{i^*-1}}|\mathbf{x})] \\ &\geq (1-\epsilon_1)(1-3\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_{\sigma_{i^*-1}}\|_1] \\ &\geq (1-2\epsilon_1)(1-3\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_{\sigma_{i^*}-1}\|_1] \\ &\geq (1-5\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_{\sigma_{i^*}-1}\|_1].\end{aligned}$$

Ta có điều phải chứng minh. □

Nếu vòng lặp bên trong bao gồm $\log(n)/\epsilon_0$ bước lặp, Bổ đề 4.5 cho thấy một tính chất quan trọng để phân tích đảm bảo hiệu suất của thuật toán.

Bổ đề 4.5. *Đặt $\mathbf{x}_i$ là $\mathbf{x}$ ở cuối bước lặp thứ i của vòng lặp ngoài. Nếu nó bao gồm $1/\epsilon_0 \log n$ vòng lặp bên trong, chúng ta có $\mathbb{E}[X] = \emptyset$ và:*

$$\alpha - \mathbb{E}[g(\mathbf{x}_i)] < \frac{\mathsf{opt}(1-5\epsilon_1)(\alpha - \mathbb{E}[g(\mathbf{x}_{i-1})])}{v}.$$

Chứng minh. Từ Bổ đề 4.2, có ít nhất ϵ_0 phần của các phần tử trong X bị loại bỏ ở mọi lần lặp của vòng lặp bên trong theo kỳ vọng. Do đó, tại bước lặp $\log(n)/\epsilon_0$, chúng ta có $\mathbb{E}[|X|] \leq n(1-\epsilon_0)^{\log(n)/\epsilon_0} < 1$. Như vậy $\mathbb{E}[X] = \emptyset$.

Nếu $X = \emptyset$, có $X_{\sigma_{i^*}} = \emptyset$ sau bước lặp thứ $\log(n)/\epsilon_0$ của vòng lặp bên trong.

Vì vậy, với bất kỳ phần tử $e \in X : \mathbf{x}_i + \chi_e \leq \mathbf{B}$ nào, ta có $g(\chi_e|\mathbf{x}_i) < \theta$, suy ra:

$$\begin{aligned}
\alpha - g(\mathbf{x}_i) &= g(\mathbf{o} \vee \mathbf{x}_i) - g(\mathbf{x}_i) \\
&\leq \sum_{e \in \{\mathbf{o} \vee \mathbf{x}_i - \mathbf{x}_i\}} g(\chi_e|\mathbf{x}_i) \\
&= \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}_i\}} g(\chi_e|\mathbf{x}_i) \\
&< \sum_{e \in \{\mathbf{o} - \mathbf{o} \wedge \mathbf{x}_i\}} \theta \\
&\leq \frac{\mathsf{opt}(1 - 5\epsilon_1)(\alpha - g(\mathbf{x}_{i-1}))}{v}.
\end{aligned}$$

Trong đó biểu thức thứ 2 theo phép toán đồng nhất $\mathbf{x} \vee \mathbf{y} - \mathbf{y} = \mathbf{x} - \mathbf{x} \wedge \mathbf{y}$ với $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_+^V$. □

Từ các Bổ đề trên, ta đưa ra được Định lý 4.1 về đảm bảo lý thuyết.

Định lý 4.1. *Với tiên đoán giá trị tối ưu v, với $(1 - 5\epsilon_1)v \leq \mathsf{opt} \leq v$, thuật toán* **AdaptDRSC** *cần $O\left((n + \log(n)\log(B)\log(1/\epsilon)/\epsilon^3)\log(1/\lambda)/\epsilon^2\right)$ truy vấn, chạy trong $O(\log(n)\log(1/\lambda)/\epsilon^2)$ vòng lặp tuần tự.*

Nó trả về một lời giải xấp xỉ tiêu chí kép đạt tỉ lệ $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$ theo kỳ vọng.

Chứng minh. Thuật toán bao gồm hai vòng lặp, do đó, ta cần phân tích độ phức tạp của truy vấn và độ phức tạp song song của từng vòng lặp. Vòng lặp bên ngoài chạy tối đa $\log(1/\lambda)/(5\epsilon_1)$ vòng tuần tự. Tại bất kỳ bước lặp nào của vòng lặp bên ngoài, có nhiều nhất $\log(n)/\epsilon_0$ bước lặp của vòng lặp bên trong.

Tại mỗi bước lặp của vòng lặp trong, thuật toán cần tính toán R_{σ_i} với $\sigma_i \in I$ để tìm σ_{i^*} (dòng 9). Thao tác này cần nhiều nhất $N\log(nB)/\epsilon_1$ phép tính hàm mục tiêu. Thuật toán sau đó tìm $X_{\sigma_{i^*}}$ trong $|X| \leq n$ phép tính hàm mục tiêu (dòng 10). Do đó hai nhiệm vụ này có thể thực hiện song song chỉ trong một vòng. Vậy độ phức tạp song song của thuật toán tối đa là:

$$\frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})\frac{1}{\epsilon_0}\log n = O(\frac{1}{\epsilon^2}\log(\frac{1}{\lambda})\log n).$$

Và số lượng truy vấn nhiều nhất là:

$$\begin{aligned}
&\frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})\sum_{l=1}^{\log(n)/\epsilon_0}\left(\frac{N}{\epsilon_1}\log(nB)+n(1-\epsilon_1)^l\right)\\
&\leq \frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})\left(\frac{\log n}{\epsilon_0}\frac{N}{\epsilon_1}\log(nB)+\frac{n}{\epsilon_0}\right)\\
&= \frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})\left(\frac{N\log^2 n}{\epsilon_0\epsilon_1}+\frac{N\log(n)\log(B)}{\epsilon_1\epsilon_0})+\frac{n}{\epsilon_1}\right)\\
&= O\left(\frac{1}{\epsilon^2}\log(\frac{1}{\lambda})(n+\frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})\log(n)\log(B))\right).
\end{aligned}$$

Để đảm bảo hiệu suất, trước tiên đặt l_t là số lần lặp của vòng lặp bên trong của bước lặp thứ t của vòng lặp ngoài. Giả sử rằng thuật toán kết thúc sau T lặp lại vòng lặp bên ngoài, ta xem xét hai trường hợp sau.

- Trường hợp 1. Nếu $T \leq \log(1/\lambda)/(5\epsilon_1)$ và $l_T < \log(n)/\epsilon_0$, thuật toán gặp điều kiện $f(\mathbf{x}_T) \geq (1-\lambda)\alpha$ tại bước lặp T và không gặp điều kiện này tại các bước lặp sớm hơn. Vì vậy, chúng ta phải có $l_t = \log(n)/\epsilon_0, \forall t < T$. Tại bước lặp thứ t của vòng lặp ngoài, đặt:

+/ $\mathbf{y}_t^l$ là $\mathbf{y}_{\sigma_{i^*}-1}$ khi kết thúc bước lặp l của vòng lặp trong.

+/ $\mathbf{x}_t^l$ là x khi kết thúc bước lặp l của vòng lặp trong và $\mathbf{x}_t^0 = \mathbf{x}_{t-1}$.

+/ $\mathbf{x}_t$ là x khi kết thúc vòng lặp ngoài.

Tại bất kỳ bước lặp $t \leq T$ của vòng lặp ngoài, sử dụng Bổ đề 4.4 cho ta:

$$\begin{aligned}
\mathbb{E}[g(\mathbf{x}_t)-g(\mathbf{x}_{t-1})] &\geq \mathbb{E}[\sum_{l=1}^{l_t}(g(\mathbf{x}_t^l)-g(\mathbf{x}_t^{l-1}))] &(4.17)\\
&\geq \sum_{l=1}^{l_t}(1-5\epsilon_1)\theta\mathbb{E}[\|\mathbf{y}_t^l\|_1] &(4.18)\\
&= (1-5\epsilon_1)\theta C_t &(4.19)
\end{aligned}$$

Trong đó $C_t = \sum_{l=1}^{l_t}\mathbb{E}[\|\mathbf{y}_t^l\|_1]$. Vì vậy:

$$\mathbb{E}[g(\mathbf{x}_t)-g(\mathbf{x}_{t-1})] \geq \frac{(1-5\epsilon_1)^2}{\upsilon}C_t(\alpha-g(\mathbf{x}_{t-1})). \quad (4.20)$$

Sắp xếp lại bất đẳng thức trên, ta có:

$$\alpha - \mathbb{E}[g(\mathbf{x}_t)] \leq (1 - \frac{(1-5\epsilon_1)^2}{v} C_t)(\alpha - g(\mathbf{x}_{t-1})). \tag{4.21}$$

Với mọi $t \leq T$. Áp dụng liên tiếp bất đẳng thức trên, ta được:

$$\begin{aligned}
\alpha\lambda &< \alpha - \mathbb{E}[g(\mathbf{x}_{T-1})] && (4.22)\\
&\leq (1 - \frac{(1-5\epsilon_1)^2}{v} C_{T-1})(\alpha - g(\mathbf{x}_{T-2})) && (4.23)\\
&\leq (1 - \frac{(1-5\epsilon_1)^3}{\mathsf{opt}} C_{T-1})(\alpha - g(\mathbf{x}_{T-2})) && (4.24)\\
&\leq e^{-\frac{(1-5\epsilon_1)^3}{\mathsf{opt}} C_{T-1}}(\alpha - g(\mathbf{x}_{T-2})) && (4.25)\\
&\leq \ldots \leq e^{-\frac{(1-5\epsilon_1)^3}{\mathsf{opt}} \sum_{l=1}^{T-1} C_l}\alpha && (4.26)\\
&\leq e^{-\frac{(1-5\epsilon_1)^3}{\mathsf{opt}} \mathbb{E}[\|\mathbf{x}_{T-1}\|_1]}\alpha. && (4.27)
\end{aligned}$$

Trong đó bất đẳng thức (4.24) là do $v \leq \mathsf{opt}/(1-5\epsilon_1)$ và bất đẳng thức (4.25) là do $e^x \geq x + 1, \forall x$. Bất đẳng thức (4.27) ngụ ý rằng

$$\mathbb{E}[c(\mathbf{x}_{T-1})] \leq \frac{\mathsf{opt}}{(1-5\epsilon_1)^3} \log \frac{1}{\lambda}.$$

Mặt khác, từ bất đẳng thức (4.20) với lưu ý rằng $g(\mathbf{x}_T) \leq \alpha$, ta có:

$$\begin{aligned}
C_T &\leq \frac{v}{(1-5\epsilon_1)^2} \frac{g(\mathbf{x}_T) - g(\mathbf{x}_{T-1})}{\alpha - g(\mathbf{x}_{T-1})} && (4.28)\\
&\leq \frac{\mathsf{opt}}{(1-5\epsilon_1)^3} \frac{g(\mathbf{x}_T) - g(\mathbf{x}_{T-1})}{\alpha - g(\mathbf{x}_{T-1})} && (4.29)\\
&\leq \frac{\mathsf{opt}}{(1-5\epsilon_1)^3}. && (4.30)
\end{aligned}$$

Như vậy:

$$\begin{aligned}\mathbb{E}[\|\mathbf{x}_T\|_1] &\leq \mathbb{E}[\|\mathbf{x}_{T-1}\|_1] + C_T \\ &\leq \frac{1}{(1-5\epsilon_1)^3}(1+\log\frac{1}{\lambda})\mathsf{opt} \\ &\leq (1+10\epsilon_1)^3(1+\log\frac{1}{\lambda})\mathsf{opt} \\ &\leq (1+50\epsilon_1)(1+\log\frac{1}{\lambda})\mathsf{opt} = (1+\epsilon)(1+\log\frac{1}{\lambda})\mathsf{opt}.\end{aligned}$$

Trong đó bất đẳng thức thứ hai do $\epsilon_1 = \epsilon/50 < 1/50$.
- Trường hợp 2. Nếu $T = \log(1/\lambda)/(5\epsilon_1)$ và $l_T = \log(n)/\epsilon_0$ ta có: $\mathbb{E}[X] = \emptyset$. Theo Bổ đề 4.5, ta có:

$$\begin{aligned}\alpha - \mathbb{E}[g(\mathbf{x}_T)] &\leq \mathsf{opt}(1-5\epsilon_1)(\alpha - \mathbb{E}[g(\mathbf{x}_{T-1})])/v \\ &\leq (1-5\epsilon_1)(\alpha - \mathbb{E}[g(\mathbf{x}_{T-1})]) \\ &\leq \ldots \leq (1-5\epsilon_1)^T\alpha \\ &\leq e^{-5\epsilon_1\frac{1}{5\epsilon_1}\log(\frac{1}{\lambda})}\alpha \\ &= \lambda\alpha.\end{aligned}$$

Do đó $\mathbb{E}[g(\mathbf{x}_T)] \geq (1-\lambda)\alpha$ ngụ ý rằng $\mathbb{E}[f(\mathbf{x}_T)] \geq (1-\lambda)\alpha$.

Mặt khác, trong trường hợp này, thuật toán không thỏa mãn điều kiện dừng. Do đó, $f(\mathbf{x}_{T-1}) < (1-\lambda)\alpha$ ngụ ý rằng $\alpha - g(\mathbf{x}_{T-1}) > \alpha\lambda$. Bằng lập luận tương tự của trường hợp 1, ta cũng có $\mathbb{E}[c(\mathbf{x}_{T-1})] \leq \log(1/\lambda)\mathsf{opt}/(1-5\epsilon_1)^3$ và $C_T \leq \mathsf{opt}/(1-5\epsilon_1)^2$. Do đó, trong trường hợp này có:

$$\mathbb{E}[\|\mathbf{x}_T\|_1] \leq (1+\epsilon)(1+\log(\frac{1}{\lambda}))\mathsf{opt}.$$

Ta có điều phải chứng minh. □

4.5.3. Thuật toán chính: BA

Từ thuật toán **AdaptDRSC**, luận án xây dựng thuật toán xấp xỉ tiêu chí kép **BA** (*Bi-criteria Algorithm*) với giả thiết opt đã biết được loại bỏ. Thuật toán chính nhận các tham số chính xác $\epsilon \in (0,1), \lambda \in (0,1)$ làm đầu vào. Vì $1 \leq \mathsf{opt} \leq Bn$, ta có thể đoán được v của opt thỏa mãn $(1-5\epsilon_1)v \leq \mathsf{opt} \leq v$, trong đó $\epsilon_1 = \epsilon/50$

bằng cách tìm kiếm trên tập

$$C = \left\{ \frac{1}{(1-5\epsilon_1)^j} : 0 \le j \le \log_{\frac{1}{1-5\epsilon_1}}(nB), j \in \mathbb{N} \right\}.$$

Đối với mỗi $v \in C$, ta tìm các lời giải ứng viên bằng cách điều chỉnh thuật toán $\mathbf{AdaptDRSC}$ và chọn giải pháp có lực lượng nhỏ nhất $\mathbf{x}$ thỏa mãn $f(\mathbf{x}) \ge (1-\lambda)\alpha$. Chi tiết đầy đủ của thuật toán chính được trình bày trong Thuật toán 12. Hiệu suất của thuật toán được nêu trong Định lý 4.2.

Algorithm 12 : $\mathbf{BA}$

Input: $f : \mathbb{Z}_+^V \mapsto \mathbb{R}_+$, a positive integer B, α, $\epsilon \in (0,1), \lambda \in (0,1)$
Output: vector $\mathbf{x}$

1: $\epsilon_1 \leftarrow \frac{\epsilon}{50}, C \leftarrow \{\frac{1}{(1-5\epsilon_1)^j} : 0 \le j \le \log_{\frac{1}{1-5\epsilon_1}}(nB)\}$
2: **for all** $v \in C$ **(in parallel) do**
3: $\quad \mathbf{x}^v \leftarrow \mathbf{AdaptDRSC}(f, B, \alpha, \epsilon_1, \lambda, v)$
4: **end for**
5: $\mathbf{x} \leftarrow \arg\min_{v \in V : f(\mathbf{x}^v) \ge (1-\lambda)\alpha} \|\mathbf{x}^v\|$
6: **return** $\mathbf{x}$

Định lý 4.2. *Với $\lambda \in (0,1)$, $\epsilon \in (0,1)$, thuật toán $\mathbf{BA}$ chạy trong $O\left(\log(1/\lambda)\log(n)/\epsilon^2\right)$ vòng lặp tuần tự. Nó tốn $O\left((n+\log(n)\log(B)\log(1/\epsilon)/\epsilon^3)\log(nB)\log(1/\lambda)/\epsilon^3\right)$ truy vấn và trả về lời giải xấp xỉ tiêu chí kép đạt tỉ lệ $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$ theo kỳ vọng.*

Chứng minh. Vòng lặp chính bao gồm nhiều nhất $\log(nB)/(5\epsilon_1)$ bước lặp và nó có thể được thực hiện song song. Độ phức tạp song song của thuật toán 12 là $O\left(\log(1/\lambda)\log(n)/\epsilon^2\right)$ và độ phức tạp của truy vấn là

$$\frac{1}{5\epsilon_1}\log(nB) \cdot O\left(\frac{1}{\epsilon^2}\log(\frac{1}{\lambda})(n + \frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})\log(n)\log(B))\right) \tag{4.31}$$

$$= O\left(\frac{1}{\epsilon^3}\log(\frac{1}{\lambda})\log(nB)(n + \frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})\log(n)\log(B))\right). \tag{4.32}$$

Tồn tại một $v \in A$ nào đó thỏa mãn $(1-5\epsilon_1)v \le \mathsf{opt} \le v$. Đặt $j' = \lceil \log_{1/(1-5\epsilon_1)}(\mathsf{opt}) \rceil$ và $v = 1/(1-5\epsilon_1)^{j'}$, ta có $v \ge \mathsf{opt}$ và

$$\log_{\frac{1}{1-5\epsilon_1}}(\mathsf{opt}) \ge \lfloor \log_{\frac{1}{1-5\epsilon_1}}(\mathsf{opt}) \rfloor = j' - 1.$$

Vì vậy:

$$\mathsf{opt} \geq \frac{1}{(1-5\epsilon_1)^{j'-1}} \geq (1-5\epsilon_1)v. \tag{4.33}$$

Bằng cách sử dụng thuật toán $\mathbf{AdaptDRSC}$, lời giải $\mathbf{x}^v$ đạt được đảm bảo hiệu suất như trong Định lý 4.1. Vì thuật toán chính trả về lời giải cho kích thước nhỏ nhất mà điều kiện chính xác được thỏa mãn. □

Lưu ý rằng ta có thể giảm số lượng truy vấn của thuật toán xuống

$$\begin{aligned} &O\left(\frac{1}{\epsilon^2}\log(\frac{1}{\lambda})\log(\frac{1}{\epsilon})\log\log(nB)(n+\frac{1}{\epsilon^3}\log(\frac{1}{\epsilon})\log(n)\log(B))\right) \\ &= O\left((n+\log(n)\log(B))\log\log(nB)\right) \end{aligned}$$

nếu sử dụng thủ tục tìm kiếm nhị phân để tìm $\mathbf{x}$. Tuy nhiên, trong trường hợp này, độ phức tạp song song tăng từ $O(\log n)$ thành $O(\log(n)\log\log(nB))$.

Đối với trường hợp của bài toán SC, tức là $B = 1$, thuật toán này vẫn đảm bảo cùng tỉ lệ xấp xỉ tiêu chí kép và độ phức tạp song song. Độ phức tạp truy vấn lúc này giảm xuống còn $O\left(n\log(n)\log(1/\lambda)/\epsilon^3\right)$. Từ Định lý 4.2, có hệ quả cho SC dưới đây.

Hệ quả 4.1. *Đối với trường hợp của bài toán SC, tức là $B = 1$, thuật toán* $\mathbf{BA}$ *có độ phức tạp song song là $O\left(\log(n)\log(1/\lambda)/\epsilon^2\right)$, cần độ phức tạp của truy vấn là $O\left(n\log(n)\log(1/\lambda)/\epsilon^3\right)$ và trả về lời giải xấp xỉ tiêu chí kép với tỉ lệ $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$ theo kỳ vọng.*

4.6. Kết luận chương

Trong chương này, luận án đề xuất một thuật toán song song hiệu quả cung cấp giải pháp xấp xỉ tiêu chí kép và cải thiện đáng kể cả độ phức tạp song song và truy vấn của các thuật toán hiện có cho các bài toán DRSC và SC.

Cho DRSC, thuật toán xấp xỉ tiêu chí kép này cho tỉ lệ $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$, với $\epsilon, \lambda \in (0,1)$ là các tham số đầu vào đã cho. Thuật toán cần $O((n+\log(n)\log(B))\log(nB))$ truy vấn và chạy với độ phức tạp song song là $O(\log n)$. Cho nên thuật toán của luận án đảm bảo độ phức tạp song song và truy vấn tốt hơn so với các thuật toán tiên tiến hiện nay.

Để giải bài toán SC, thuật toán cho lời giải xấp xỉ đạt tiêu chí kép $((1+\epsilon)(1+\log(1/\lambda)), 1-\lambda)$ với độ phức tạp song song là $O(\log n)$ và độ phức tạp truy vấn là $O(n\log n)$.

Các đảm bảo lý thuyết trên đây đều có giá trị và vượt trội hơn so với các thuật toán xấp xỉ hiện nay. Mặt khác, thuật toán này cũng có thể phát triển trong tương lai khi đưa thêm các ràng buộc như chi phí, matroid... giải trên lưới nguyên.

Các kết quả nghiên cứu của luận án đã được công nhận tại bài báo *"A note for approximating the submodular cover problem over integer lattice with low adaptive and query complexities"*, tạp chí Information Processing Letter, Tập 182, 2023 (**ISI/Q3**).

KẾT LUẬN

Luận án nghiên cứu thuật toán xấp xỉ cho bài toán tối ưu tổ hợp với dữ liệu lớn, trong đó tập trung nghiên cứu một số bài toán thường gặp trong phân tích dữ liệu. Từ đó, khái quát thành bài toán tối đa hàm submodular có ràng buộc và các bài toán mở rộng khác. Cụ thể, các bài toán mà luận án đã trình bày gồm: Bài toán tối đa hàm k-submodular với ràng buộc chi phí - kSMK; Bài toán tối đa hàm submodular với ràng buộc chi phí có nhiễu - SMKN; Bài toán Phủ Submoudlar trên lưới nguyên - DRSC. Với mỗi toán, luận án đưa ra các thuật toán xấp xỉ giải quyết bài toán cho tỉ lệ xấp xỉ cạnh tranh, giảm độ phức tạp truy vấn hoặc độ phức tạp song song, qua đó góp phần giảm thời gian chạy. Các đóng góp của luận án bao gồm:

1. Đối với bài toán kSMK, luận án đề xuất thuật toán **FA** đưa độ phức tạp truy vấn về tuyến tính, trên cơ sở đó đề xuất thuật toán cải tiến **IFA**, cải tiến tăng cường **IFA+** giải quyết bài toán với trường hợp hàm mục tiêu đơn điệu. **IFA+** cho tỉ lệ xấp xỉ tốt hơn so với thuật toán tốt nhất hiện nay, với độ phức tạp truy vấn giảm xuống một hệ số. Luận án cũng mở rộng kết quả nghiên cứu giải quyết với hàm mục tiêu không đơn điệu, đề xuất thuật toán xấp xỉ tuyến tính tăng cường **RLA**, cho tỉ lệ xấp xỉ tốt tương đương với thuật toán tốt nhất hiện nay nhưng độ phức tạp truy vấn cũng giảm xuống một hệ số.

2. Đối với bài toán SMKN, luận án đề xuất thuật toán tham lam **GUN** giải quyết bài toán. Để cải thiện tốc độ của thuật toán và không gian lưu trữ, luận án đã đề xuất thuật toán luồng **NS** để giải bài toán này.

3. Đối với bài toán DRSC, luận án xây dựng một thuật toán song song tiêu chí kép **BA** cho độ phức tạp truy vấn và độ phức tạp song song tốt hơn hẳn các thuật toán tốt nhất hiện nay. Các kết quả có thể áp dụng được sang bài toán SC và cho kết quả vượt trội hơn các kết quả hiện nay.

Trong tương lai, NCS tiếp tục mở rộng nghiên cứu các bài toán biến thể khác của bài toán tối đa hàm submodular. Các vấn đề có thể mở rộng nghiên cứu:

1. Tối đa hàm submodular với ràng buộc d-knap;
2. Tối đa hàm non-submodular;
3. Bài toán kSMK trong môi trường nhiễu;
4. Các bài toán SM giải trên lưới nguyên.

Các hướng nghiên cứu bao gồm:

- Tiếp tục phát triển các thuật toán hiệu quả cho các bài toán;
- Nghiên cứu các ứng dụng trong học máy và trí tuệ nhân tạo là thể hiện của các bài toán tối đa hàm submodular.

DANH MỤC CÔNG TRÌNH KHOA HỌC CỦA TÁC GIẢ LIÊN QUAN ĐẾN LUẬN ÁN

1. **Dung T. K Ha**, Canh V. Pham and Huan X. Hoang, *Submodular Maximization Subject to a Knapsack Constraint Under Noise Models*, Asia-Pacific Journal of Operational Research, Volume 39, Number 6, 2022 (**ISI/Q3**);

2. Canh V. Pham and **Dung T.K. Ha**. *A note for approximating the submodular cover problem over integer lattice with low adaptive and query complexities*. Information Processing Letters, Volume 182, 2023 (**ISI/Q3**).

3. Canh V Pham, **Dung KT Ha**, Huan X Hoang and Tan D Tran, *Fast Streaming Algorithms for k-Submodular Maximization under a Knapsack Constraint*, IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA), 2022 (**ISI/RANK A**);

4. **Dung T.K. Ha**, Canh V. Pham, Tan D. Tran, *Improved Approximation Algorithms for k-Submodular Maximization under a Knapsack Constraint*, Computers & Operations Research, 2023 (**ISI/Q1**);

5. **Dung T. K Ha**, Canh V. Pham, Tan D. Tran and Huan X. Hoang. *Robust Approximation Algorithms for Non-monotone k-Submodular Maximization under a Knapsack Constraint*, The 15th IEEE International Conference on KNOWLEDGE AND SYSTEMS ENGINEERING (KSE 2023);

TÀI LIỆU THAM KHẢO

[1] G. Amanatidis, F. Fusco, P. Lazos, S. Leonardi, A. Marchetti-Spaccamela, and R. Reiffenhäuser. Submodular maximization subject to a knapsack constraint: Combinatorial algorithms with near-optimal adaptive complexity. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 231–242. PMLR.

[2] G. Amanatidis, F. Fusco, P. Lazos, S. Leonardi, and R. Reiffenhäuser. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Advances in Neural Information Processing Systems*, volume 33, pages 16903–16915, 2020.

[3] A. P. S. Andreas Krause and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[5] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: massive data summarization on the fly. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 671–680. ACM, 2014.

[6] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 1497–1514. SIAM, 2014.

[7] E. Balkanski, A. Breuer, and Y. Singer. Non-monotone submodular maximization in exponentially fewer iterations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pages 2359–2370, 2018.

[8] E. Balkanski, S. Qian, and Y. Singer. Instance specific approximations for submodular maximization. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139 of *Proceedings of Machine Learning Research*, pages 609–618. PMLR, 2021.

[9] E. Balkanski, A. Rubinstein, and Y. Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 283–302. SIAM, 2019.

[10] E. Balkanski, A. Rubinstein, and Y. Singer. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 66–77. ACM, 2019.

[11] E. Balkanski and Y. Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018,*, pages 1138–1151. ACM, 2018.

[12] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *Internet and Network Economics, Third International Workshop, WINE*, 2007.

[13] J. A. Bilmes. Submodularity in machine learning and artificial intelligence. *CoRR*, abs/2202.00132, 2022.

[14] G. E. Blelloch, R. Peng, and K. Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 23–32. ACM, 2011.

[15] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux. Intel lab sensor, 2004.

[16] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 946–957, 2014.

[17] A. Borodin, Y. Filmus, and J. Oren. Threshold models for competitive influence in social networks. In *Internet and Network Economics - 6th International Workshop, WINE 2010*, pages 539–550, 2010.

[18] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), 2001 - Volume 1*, pages 105–112. IEEE Computer Society, 2001.

[19] A. Breuer, E. Balkanski, and Y. Singer. The FAST algorithm for submodular maximization. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1134–1143. PMLR, 2020.

[20] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 649–658, 2012.

[21] N. Buchbinder, M. Feldman, and R. Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1149–1168. SIAM, 2015.

[22] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference*, volume 4513 of *Lecture Notes in Computer Science*, pages 182–196. Springer.

[23] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[24] D. T. K. H. Canh V. Pham, Tan D. Tran and M. T. Thai. Linear query approximation algorithms for non-monotone submodular maximization under knapsack constraint. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 4493–4501, 2023.

[25] H. X. H. Canh V. Pham, Hieu V. Duong and M. T. Thai. Competitive influence maximization within time and budget constraints in online social networks: An algorithmic approach. *Applied Sciences*, 9(11), 2019.

[26] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[27] C. Chekuri and K. Quanrud. Submodular function maximization in parallel via the multilinear relaxation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 303–322, 2019.

[28] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 43(6):1831–1879, 2014.

[29] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincón, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence maximization in social networks when negative opinions may emerge and propagate. In *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011*, pages 379–390, 2011.

[30] W. Chen, L. V. S. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.

[31] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2010*, pages 1029–1038, 2010.

[32] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2009*, pages 199–208. ACM, 2009.

[33] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM 2010, The 10th IEEE International Conference on Data Mining*, pages 88–97, 2010.

[34] Y. Chen, T. Dey, and A. Kuhnle. Best of both worlds: Practical and theoretically optimal submodular maximization in parallel. In *Advances in Neural Information Processing Systems*, volume 34, pages 25528–25539, 2021.

[35] Y. Chen and A. Kuhnle. Approximation algorithms for size-constrained non-monotone submodular maximization in deterministic linear time. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023*, pages 250–261, 2023.

[36] E. Cohen, D. Delling, T. Pajor, and R. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management 2014*, pages 629–638.

[37] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley, 1998.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2009.

[39] V. G. Crawford. An efficient evolutionary algorithm for minimum cost submodular cover. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 1227–1233, 2019.

[40] V. G. Crawford, A. Kuhnle, and M. T. Thai. Submodular cost submodular cover with an approximate oracle. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 1426–1435. PMLR, 2019.

[41] S. Cui, K. Han, J. Tang, H. Huang, X. Li, and Z. Li. Streaming algorithms for constrained submodular maximization. In *Abstract Proceedings of the 2023 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2023*, pages 65–66, 2023.

[42] G. Das, C. M. Jermaine, and P. A. Bernstein, editors. *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018*. ACM, 2018.

[43] J. M. K. David Kempe and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2003*, pages 137–146, 2003.

[44] T. Dey, Y. Chen, and A. Kuhnle. DASH: A distributed and parallelizable algorithm for size-constrained submodular maximization. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023*, pages 3941–3948, 2023.

[45] D.-Z. Du, P. Pardalos, X. Hu, and W. Wu. *Introduction to Combinatorial Optimization*. 01 2022.

[46] W. Du, Z. Xing, M. Li, B. He, L. H. C. Chua, and H. Miao. Optimal sensor placement and measurement of wind for water quality studies in urban reservoirs. In *IPSN'14, Proceedings of the 13th International Symposium on Information Processing in Sensor Networks (part of CPS Week)*, pages 167–178, 2014.

[47] A. Ene and H. L. Nguyen. A reduction for optimizing lattice submodular functions with diminishing returns. *CoRR*, abs/1606.08362, 2016.

[48] A. Ene and H. L. Nguyen. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPIcs*, pages 53:1–53:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[49] A. Ene and H. L. Nguyen. Parallel algorithm for non-monotone dr-submodular maximization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2902–2911, 2020.

[50] A. Ene, H. L. Nguyen, and A. Vladu. Submodular maximization with matroid and packing constraints in parallel. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 90–101. ACM, 2019.

[51] M. Fahrbach, V. S. Mirrokni, and M. Zadimoghaddam. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1833–1842, 2019.

[52] M. Fahrbach, V. S. Mirrokni, and M. Zadimoghaddam. Submodular maximization with nearly optimal approximation, adaptivity and query complexity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 255–273. SIAM, 2019.

[53] Y. Fairstein, A. Kulik, J. S. Naor, D. Raz, and H. Shachnai. An almost optimal approximation algorithm for monotone submodular multiple knapsack. *Journal of Computer and System Sciences*, 125:149–165, 2022.

[54] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[55] U. Feige, V. S. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[56] M. Feldman, J. Naor, and R. Schwartz. Nonmonotone submodular maximization via a structural continuous greedy algorithm-(extended abstract). In *Proc. of the 38th ICALP 2011*, volume 6755, pages 342–353, 2011.

[57] S. Fujishige. Algorithms for solving the independent-flow problems. *Journal of the Operations Research Society of Japan*, 21(2):189–204, 1978.

[58] S. Fujishige. *Submodular systems and related topics*, pages 113–131. Springer Berlin Heidelberg, 1984.

[59] S. Fujishige. *Submodular Functions and Optimization*, volume 58 of *Annals of Discrete Mathematics*, page 1. Elsevier, 2005.

[60] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

[61] P. Gölz and A. D. Procaccia. Migration as submodular optimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 549–556. AAAI Press, 2019.

[62] R. Gomes and A. Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 391–398. Omnipress, 2010.

[63] A. Goyal, F. Bonchi, L. V. S. Lakshmanan, and S. Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Network Analysis and Mining*, 3(2):179–192, 2013.

[64] A. Guillory and J. A. Bilmes. Simultaneous learning and covering with adversarial noise. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 369–376, 2011.

[65] A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. *CoRR*, abs/1003.1517, 2010.

[66] R. Haba, E. Kazemi, M. Feldman, and A. Karbasi. Streaming submodular maximization under a k-set system constraint. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 3939–3949, 2020.

[67] K. Han, S. Cui, T. Zhu, E. Zhang, B. Wu, Z. Yin, T. Xu, S. Tang, and H. Huang. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(1):05:1–05:31.

[68] J. D. Hartline, V. S. Mirrokni, and M. Sundararajan. Optimal marketing strategies over social networks. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, pages 189–198. ACM, 2008.

[69] T. Horel and Y. Singer. Maximization of approximately submodular functions. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 3045–3053, 2016.

[70] C. Huang, N. Kakimura, and Y. Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 82(4):1006–1032, 2020.

[71] M. T. T. Hung T. Nguyen and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016*, pages 695–710, 2016.

[72] D. V. Huy and N. D. Viet. Df-ams: Proposed solutions for multi-sensor data fusion in wireless sensor networks. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–6, 2015.

[73] S. Iwata, S. Tanigawa, and Y. Yoshida. Improved approximation algorithms for k-submodular function maximization. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 404–413, 2016.

[74] W. J. Kaiser, G. J. Pottie, M. B. Srivastava, G. S. Sukhatme, J. D. Villasenor, and D. Estrin. Networked infomechanical systems (NIMS) for ambient intelligence. In *Ambient Intelligence*, pages 83–113. Springer, 2005.

[75] Y. Kawahara, K. Nagano, and Y. Okamoto. Submodular fractional programming for balanced clustering. *Pattern Recognition Letters*, 32(2):235–243, 2011.

[76] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[77] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS 2004 - First Conference on Email and Anti-Spam*, 2004.

[78] P. Kohli, M. P. Kumar, and P. H. S. Torr. P3 & beyond: Move making algorithms for solving higher order functions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1645–1656, 2009.

[79] B. Korte and J. Vygen. *NP-Completeness*, pages 385–421. Springer Berlin Heidelberg, 2018.

[80] A. Krause and D. Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, February 2014.

[81] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *National Conference on Artificial Intelligence (AAAI), Nectar track*, July 2007.

[82] A. Krause and C. Guestrin. Optimizing sensing: From water to the web. *IEEE Computer*, 42(8):38–45, 2009.

[83] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Transactions on Sensor Networks*, 7(4), 2011.

[84] A. Krause, A. P. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficientalgorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[85] A. Kuhnle. Quick streaming algorithms for maximization of monotone submodular functions in linear time. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021*, volume 130 of *Proceedings of Machine Learning Research*, pages 1360–1368. PMLR.

[86] A. Kuhnle. Nearly linear-time, parallelizable algorithms for non-monotone submodular maximization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, 2021.

[87] A. Kuhnle, J. D. Smith, V. G. Crawford, and M. T. Thai. Fast maximization of non-submodular, monotonic functions on the integer lattice. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2791–2800. PMLR, 2018.

[88] A. Kulik, H. Shachnai, and T. Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics Operations Research*, 38(4):729–739, 2013.

[89] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13*, pages 1–10. ACM, 2013.

[90] L. Kumari and J. A. Bilmes. Submodular span, with applications to conditional data summarization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021*, pages 12344–12352. AAAI Press, 2021.

[91] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *TWEB*, 1(1):5, 2007.

[92] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429, 2007.

[93] J. Leskovec and Krevl. A. snap datasets: Stanford large network dataset collection, 2014.

[94] H. Lin and J. A. Bilmes. A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 510–520. The Association for Computer Linguistics, 2011.

[95] L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art, XIth International Symposium on Mathematical Programming*, pages 235–257. Springer, 1982.

[96] M. Z. M. Mitrovic, E. Kazemi and A. Karbasi. Data summarization at scale: A two-stage submodular approach. In *ICML*, page 3593–3602, 2016.

[97] B. Mirzasoleiman, A. Badanidiyuru, and A. Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1358–1367. JMLR.org, 2016.

[98] B. Mirzasoleiman, A. Karbasi, A. Badanidiyuru, and A. Krause. Distributed submodular cover: Succinctly summarizing massive data. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 2881–2889, 2015.

[99] B. Mirzasoleiman, M. Zadimoghaddam, and A. Karbasi. Fast distributed submodular cover: Public-private data summarization. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3594–3602, 2016.

[100] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[101] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

[102] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[103] D. T. Nguyen, H. Zhang, S. Das, M. T. Thai, and T. N. Dinh. Least cost influence in multiplex social networks: Model representation and analysis. In *2013 IEEE 13th International Conference on Data Mining 2013*, 2013.

[104] H. Nguyen and R. Zheng. On budgeted influence maximization in social networks. *IEEE Journal of Selected Areas in Communications*, 31(6):1084–1094, 2013.

[105] H. T. Nguyen, M. T. Thai, and T. N. Dinh. A billion-scale approximation algorithm for maximizing benefit in viral marketing. *IEEE/ACM Transactions on Networking (TON)*, 25(4):2419–2429, 2017.

[106] L. Nguyen and M. Thai. Streaming k-submodular maximization under noise subject to size constraint. In *Proceedings of the International Conference on Machine Learning , (ICML-2020), Thirty-seventh International Conference on Machine Learning*, 2020.

[107] A. Norouzi-Fard, A. Bazzi, I. Bogunovic, M. E. Halabi, Y. Hsieh, and V. Cevher. An efficient streaming algorithm for the submodular cover problem. In *Advances in Neural Information Processing Systems 29*, pages 4493–4501, 2016.

[108] A. Norouzi-Fard, J. Tarnawski, S. Mitrovic, A. Zandieh, A. Mousavifar, and O. Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3826–3835. PMLR, 2018.

[109] N. Ohsaka and Y. Yoshida. Monotone k-submodular function maximization with size constraints. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 694–702, 2015.

[110] H. Oshima. Derandomization for k-submodular maximization. In *Combinatorial Algorithms - 28th International Workshop, IWOCA 2017*, volume 10765 of *Lecture Notes in Computer Science*, pages 88–99, 2017.

[111] S. A. P. Parambath, N. Vijayakumar, and S. Chawla. SAGA: A submodular greedy algorithm for group recommendation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 3900–3908. AAAI Press, 2018.

[112] C. V. Pham, H. V. Duong, and M. T. Thai. Importance sample-based approximation algorithm for cost-aware targeted viral marketing. In *Computational Data and Social Networks - 8th International Conference, CSoNet 2019*, pages 120–132, 2019.

[113] C. V. Pham, D. V. Pham, B. Q. Bui, and A. V. Nguyen. Minimum budget for misinformation detection in online social networks with provable guarantees. *Optimization Letters*, pages 1–30, 2021.

[114] C. V. Pham, Q. V. Phu, H. X. Hoang, J. Pei, and M. T. Thai. Minimum budget for misinformation blocking in online social networks. *Journal of Combinatorial Optimization*, 38(4):1101–1127, 2019.

[115] C. V. Pham, Q. C. Vu, D. K. Ha, T. T. Nguyen, and N. D. Le. Maximizing k-submodular functions under budget constraint: applications and streaming algorithms. *Journal of Combinatorial Optimization*, 44(1):723–751, 2022.

[116] C. Qian, J. Shi, K. Tang, and Z. Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22(4):595–608, 2018.

[117] C. Qian, J. Shi, Y. Yu, K. Tang, and Z. Zhou. Subset selection under noise. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 3560–3570, 2017.

[118] A. Rafiey and Y. Yoshida. Fast and private submodular and k-submodular functions maximization with matroid constraints. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 7887–7897. PMLR, 2020.

[119] Y. Ran, Z. Zhang, and S. Tang. Improved parallel algorithm for minimum cost submodular cover problem. In *Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 3490–3502. PMLR, 2022.

[120] G. R. Rishabh Iyer. Submodular optimization approaches towards data and feature subset selection. In *Combinatorial Approaches for Data Feature Topic Selection and Summarization - IJCAI 2020 tutorial*, 2020.

[121] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 08 2007.

[122] S. Sakaue. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discrete Optimization*, 23:105–113, 2017.

[123] A. Schrijver. A course in combinatorial optimization. 03 2003.

[124] Y. Singer and A. Hassidim. Optimization for approximate submodularity. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 394–405, 2018.

[125] A. P. Singh, A. Guillory, and J. A. Bilmes. On bisubmodular maximization. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012*, volume 22 of *JMLR Proceedings*, pages 1055–1063, 2012.

[126] T. Soma. No-regret algorithms for online k-submodular maximization. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019*, volume 89 of *Proceedings of Machine Learning Research*, pages 1205–1214, 2019.

[127] T. Soma, N. Kakimura, K. Inaba, and K. Kawarabayashi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 351–359. JMLR.org.

[128] T. Soma and Y. Yoshida. A generalization of submodular cover via the diminishing return property on the integer lattice. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 847–855, 2015.

[129] T. Soma and Y. Yoshida. Maximizing monotone submodular functions over the integer lattice. *Mathematics Programming*, 172(1-2):539–563, 2018.

[130] X. Sun, J. Zhang, and Z. Zhang. Tight algorithms for the submodular multiple knapsack problem. *CoRR*, abs/2003.11450, 2020.

[131] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[132] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data 2015*, pages 1539–1554, 2015.

[133] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 75–86, 2014.

[134] Z. Tang, C. Wang, and H. Chan. On maximizing a monotone k-submodular function under a knapsack constraint. *Operations Research Letters*, 50(1):28–31, 2022.

[135] J. Thapper and S. Zivný. The power of linear programming for valued csps. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 669–678. IEEE Computer Society, 2012.

[136] A. L. Traud, P. J. Mucha, and M. A. Porter. Social structure of Facebook networks. *Phys. A*, 391(16):4165–4180, Aug 2012.

[137] S. Tschiatschek, R. K. Iyer, H. Wei, and J. A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 1413–1421, 2014.

[138] V. V. Vazirani. *Approximation algorithms*. Springer, 2004.

[139] P. Wan, D. Du, P. M. Pardalos, and W. Wu. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Applications*, 45(2):463–474, 2010.

[140] B. Wang and H. Zhou. Multilinear extension of k-submodular functions. *CoRR*, abs/2107.07103, 2021.

[141] B. Wang and H. Zhou. Multilinear extension of k-submodular functions. *CoRR*, abs/2107.07103, 2021.

[142] L. Wang, S. Ermon, and J. Hopcroft. Feature-enhanced probabilistic models for diffusion network inference. volume 7524, pages 499–514, 09 2012.

[143] Z. Wang, E. Chen, Q. Liu, Y. Yang, Y. Ge, and B. Chang. Maximizing the coverage of information propagation in social networks. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 2104–2110. AAAI Press, 2015.

[144] J. Ward and S. Zivný. Maximizing bisubmodular and k-submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, pages 1468–1481. SIAM, 2014.

[145] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[146] L. A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.

[147] R. Yang, D. Xu, Y. Cheng, C. Gao, and D. Du. Streaming submodular maximization under noises. In *39th IEEE International Conference on Distributed Computing Systems, ICDCS 2019*, pages 348–357. IEEE, 2019.

[148] Q. Yu, E. L. Xu, and S. Cui. Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations. In *2016 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2016, Washington, DC, USA, December 7-9, 2016*, pages 1295–1299. IEEE, 2016.

[149] Q. Yu, E. L. Xu, and S. Cui. Streaming algorithms for news and scientific literature recommendation: Monotone submodular maximization with a d-knapsack constraint. *IEEE Access*, 6:53736–53747, 2018.

[150] H. Zhang, H. Zhang, X. Li, and M. T. Thai. Limiting the spread of misinformation while effectively raising awareness in social networks. In *Computational Social Networks - 4th International Conference, CSoNet 2015*, pages 35–47, 2015.

[151] Y. Zhang, M. Li, D. Yang, and G. Xue. A budget feasible mechanism for k-topic influence maximization in social networks. In *2019 IEEE Global Communications Conference, GLOBECOM 2019*, pages 1–6. IEEE, 2019.

[152] L. Zheng, H. Chan, G. Loukides, and M. Li. Maximizing approximately k-submodular functions. In *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021*, pages 414–422, 2021.